Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6)
11th Meeting: Munich, DE, 15-19 March, 2004~~10th Meeting: Waikoloa, Hawaii, USA, 8-12 December, 2003~~

Document: JVT-K~~J~~0xx~~47~~
Filename: FRExtr0.doc

Title: **Draft Text of H.264/AVC ~~Professional~~ Fidelity Range Extensions Amendment**

Status: Output Document Approved by JVT

Purpose: Draft (Study text status in ISO/IEC)

Author(s) or Contact(s):

Gary Sullivan
One Microsoft Way
Redmond, WA 98053 USA
Tel: +1 (425) 703-5308
Email: garysull@microsoft.com

Tom McMahon
Dolby Laboratories, Inc.
3601 W. Alameda Ave.
Burbank, CA 91505-5300
Tel: +1 (818) 823-2855
Email: tom@dolby.com

Thomas Wiegand
Heinrich Hertz Institute (FhG),
Einsteinufer 37, D-10587 Berlin,
Germany
Tel: +49 - 30 - 31002 617
Email: wiegand@hhi.de

Source: JVT

TWIN TEXT PUBLICATION

ADVANCED VIDEO CODING

AMENDMENT 1
~~Professional~~ Fidelity Range Extensions

ITU-T Recommendation H.264 | ISO/IEC 14496-10

CONTENTS

Page

Introduction

This document is the ~~Professional~~ Fidelity Range Extensions Amendment to ITU-T Rec. H.264 | ISO/IEC 14496-10.

The Recommendation | International Standard has been modified in areas where extended sample bit depth or alternative chroma format support requires new or substitute text in order to implement an extended-capability encoder or decoder. The original Recommendation | International Standard specified operation with 8 bits per sample and 4:2:0 chroma format (a format in which the two chroma arrays each have half the horizontal and vertical resolution of the corresponding luma array). This amendment extends this capability to support up to 12 bits per sample and additional chroma formats as follows.

- Monochrome (a format in which only a luma array is present)
- 4:2:2 (a format in which the two chroma arrays each have half the horizontal resolution and the same vertical resolution as in the corresponding luma array)
- 4:4:4 (a format in which the two chroma arrays each have the same horizontal and vertical resolution as in the corresponding luma array)

The variety of color spaces that can be explicitly indicated as the source of the video content has also been extended.

A switchable 8x8 transform and a quant~~ization~~isation matrix has been added for improved coding efficiency.

Additionally, a new SEI message has been added to provide a character~~ization~~isation of film grain for use in high-quality encoding of film and video, for use when it is advantageous to synthesize film grain with specified characteristics as a post-process after decoding.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

ADVANCED VIDEO CODING

AMENDMENT 1
~~Professional~~ Fidelity Range Extensions

1) Subclause 5.7 "Mathematical Functions"

Replace Equation 5-3 of subclause 5.7 with the following two equations.

$$Clip1_Y(x) = Clip3(0, (1 << BitDepth_Y) – 1 , x) \quad (5\text{-}3_Y)$$

$$Clip1_C(x) = Clip3(0, (1 << BitDepth_C) – 1 , x) \quad (5\text{-}3_C)$$

2) Subclause 6.2 "Source, decoded, and output picture formats"

Substitute the following for the entire content of subclause 6.2.

This subclause specifies the relationship between source and decoded frames and fields that is given via the bitstream.

The video source that is represented by the bitstream is a sequence of either or both frames or fields (called collectively pictures) in decoding order.

The source and decoded pictures (frames or fields) are each comprised of one or more sample arrays:

- Luma only (monochrome)
- Luma and two Chroma (YCbCr)
- RGB, XYZ, etc.

[Ed. Specifics of RGB and XYZ handling are TBD.]

The variable ChromaFormatFactor is specified in Table 6-1, depending on the chroma format sampling structure. If not otherwise specified the value of ChromaFormatFactor shall be inferred equal to 1.5, indicating 4:2:0 sampling.

Table 6-1 – ChromaFormatFactor values

chroma_format_idc	**Chroma Format**	ChromaFormatFactor	Chroma Format Component Order List
0	monochrome	1	Luma only
1	4:2:0	1.5	Luma, Cb, Cr
2	4:2:2	2	Luma, Cb, Cr
3	4:4:4	3	Luma-Equiv (Y or G), Cb Equiv (Z or B), Cr Equiv (X or R)
4-7	Reserved	Reserved	

In monochrome sampling there is only one sample array, which shall nominally be considered a luma array.

In 4:2:0 sampling, each of the two chroma arrays has half the height and half the width of the luma array.

In 4:2:2 sampling, each of the two chroma arrays has the same height and half the width of the luma array.

In 4:4:4 sampling, each of the two chroma arrays has the same height and width as the luma array.

In 4:4:4 RGB or XYZ sampling, the G or Y array shall be considered the luma array and the R, B, X or Z arrays shall be considered the chroma arrays.

The width and height of the luma sample arrays are each a multiple of 16. In bitstreams using 4:2:0 chroma sampling, the width and height of chroma sample arrays are each an integer multiple of 8. In bitstreams using 4:2:2 sampling, the width of the chroma sample arrays is an integer multiple of 8 and the height is an integer multiple of 16. The height of a luma array that is coded as two separate fields or in macroblock-adaptive frame-field coding (see below) is an integer multiple of 32 samples. In bitstreams using 4:2:0 chroma sampling, the height of each chroma array that is coded as two separate fields or in macroblock-adaptive frame-field coding (see below) is an integer multiple of 16 samples. The width or height of pictures output from the decoding process need not be an integer multiple of 16 and can be specified using a cropping rectangle.

The width of fields coded referring to a specific sequence parameter set is the same as that of frames coded referring to the same sequence parameter set (see below). The height of fields coded referring to a specific sequence parameter set is half that of frames coded referring to the same sequence parameter set (see below).

In bitstreams of chroma_format_idc value equal to one, the nominal vertical and horizontal relative locations of luma and chroma samples in frames are shown in Figure 6-1. Alternative chroma sample relative locations may be indicated in video usability information (see Annex E).

The number of bits necessary for the representation of each of the samples in the luma and chroma arrays in a video sequence is in the range of 8 to 12, and the number of bits used in the luma array may differ from the number of bits used in the chroma arrays.

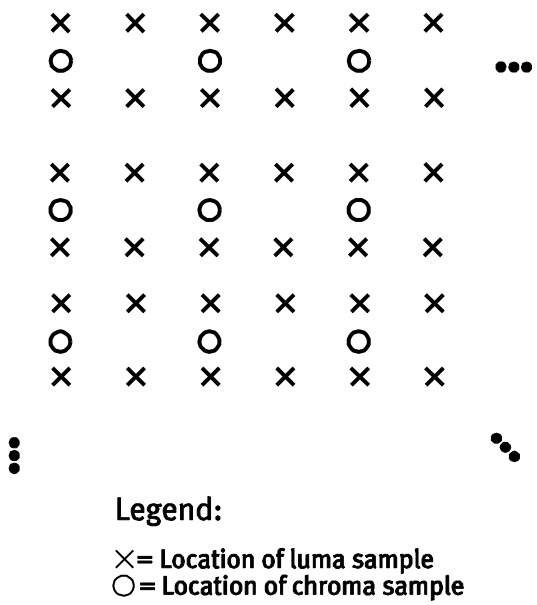


Figure 6-1 – Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a frame

A frame consists of two fields as described below. A coded picture may represent a coded frame or an individual coded field. A coded video sequence conforming to this Recommendation | International Standard may contain arbitrary combinations of coded frames and coded fields. The decoding process is also specified in a manner that allows smaller regions of a coded frame to be coded either as a frame or field region, by use of macroblock-adaptive frame-field coding.

Source and decoded fields are one of two types: top field or bottom field. When two fields are output at the same time, or are combined to be used as a reference frame (see below), the two fields (which shall be of opposite parity) are interleaved. The first (i.e., top), third, fifth, etc. rows of a decoded frame are the top field rows. The second, fourth, sixth, etc. rows of a decoded frame are the bottom field rows. A top field consists of only the top field rows of a decoded frame. When the top field or bottom field of a decoded frame is used as a reference field (see below) only the even rows (for a top field) or the odd rows (for a bottom field) of the decoded frame are used.

In bitstreams of chroma_format_idc value equal to one, the nominal vertical and horizontal relative locations of luma and chroma samples in top and bottom fields are shown in Figure 6-2. The nominal vertical sampling relative locations of the chroma samples in a top field are specified as shifted up by one-quarter luma sample height relative to the field-sampling grid. The vertical sampling locations of the chroma samples in a bottom field are specified as shifted down by one-quarter luma sample height relative to the field-sampling grid. Alternative chroma sample relative locations may be indicated in the video usability information (see Annex E).

NOTE – The shifting of the chroma samples is in order for these samples to align vertically to the usual location relative to the full-frame sampling grid as shown in Figure 6-1.

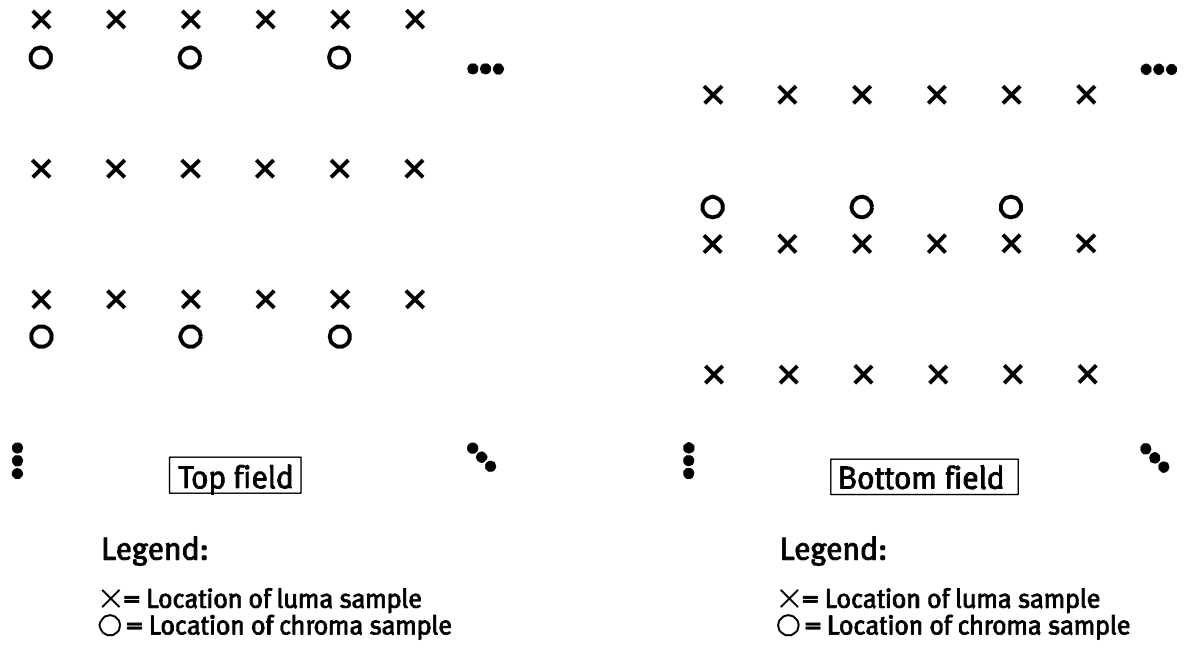


Figure 6-2 – Nominal vertical and horizontal sampling locations of 4:2:0 samples in top and bottom fields.

In bitstreams of chroma_format_idc value equal to 2, the luma and chroma samples are co-sited and the nominal locations in a frame and in fields are as shown in Figures 6-2a and 6-2b respectively.

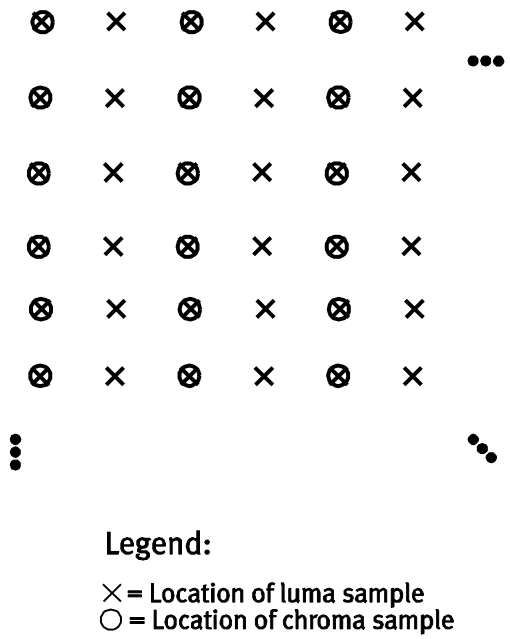


Figure 6-2a – Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a frame

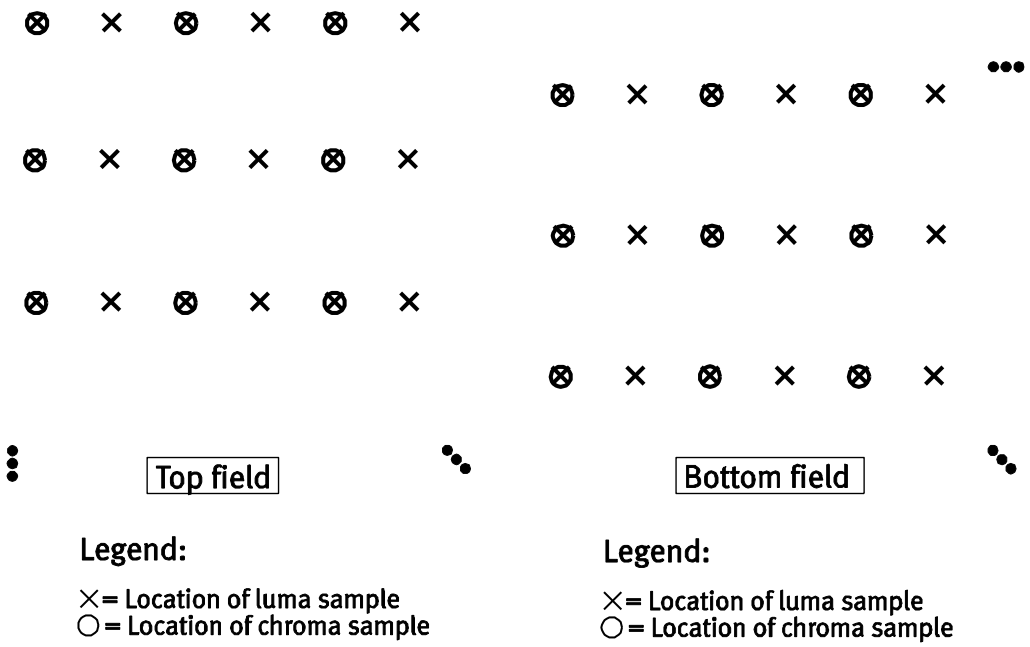


Figure 6-2b – Nominal vertical and horizontal sampling locations of 4:2:2 samples top and bottom fields.

[Ed. Note: Avoid renumbering existing figures]

In bitstreams of chroma_format_idc value equal to 3, all array samples are co-sited for all cases of frames and fields. [Ed. Note: Add a figure.]

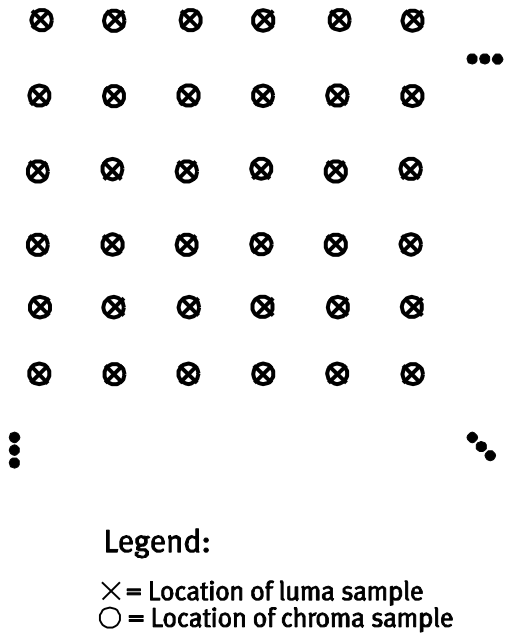


Figure 6-2c – Nominal vertical and horizontal locations of 4:4:4 luma and chroma samples in a frame

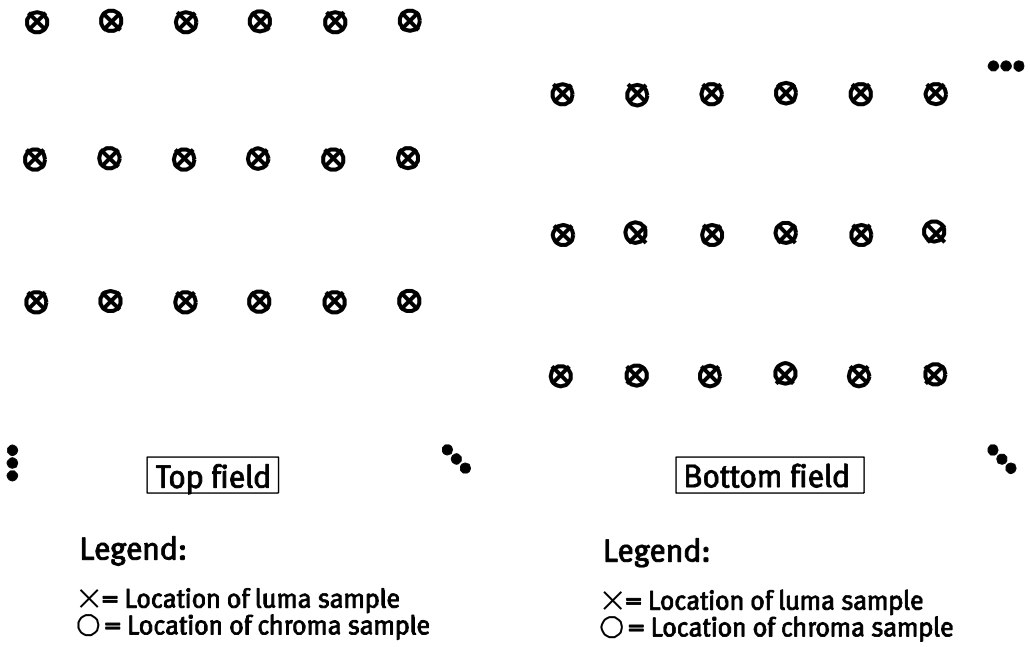


Figure 6-2d – Nominal vertical and horizontal sampling locations of 4:4:4 samples top and bottom fields
[Ed. Necessary? Will MBAFF or coded fields be supported for 4:4:4?]

[Ed. Note: Other things relating to chroma in clause 6 need work (e.g., spatial location and neighbour definitions for chroma).]

"

3) Subclause 7.3.2.1 "Sequence parameter set RBSP syntax"

Revise subclause 7.3.2.1 as follows.

seq_parameter_set_rbsp( ) {	C	Descriptor
profile_idc	0	u(8)
constraint_set0_flag	0	u(1)
constraint_set1_flag	0	u(1)
constraint_set2_flag	0	u(1)
constraint_set3_flag	0	u(1)
~~**constraint_set4_flag**~~	~~0~~	~~u(1)~~
reserved_zero_4~~3~~bits /* equal to 0 */	0	u(3)
level_idc	0	u(8)
seq_parameter_set_id	0	ue(v)
if((profile_idc ~~= = 69 || profile_idc == 70 || profile_idc == 71 || profile_idc == 72~~% 11)~~)~~ != 0)~~ {~~***~~)~~ {		
chroma_format_idc	0	u(2)
bit_depth_luma_minus8	0	ue(v)
bit_depth_chroma_minus8	0	ue(v)
~~—~~lossless~~_coding~~_qp0_flag	0	u(1)
}		
log2_max_frame_num_minus4	0	ue(v)
pic_order_cnt_type	0	ue(v)
if(pic_order_cnt_type = = 0)		
log2_max_pic_order_cnt_lsb_minus4	0	ue(v)
else if(pic_order_cnt_type = = 1) {		
delta_pic_order_always_zero_flag	0	u(1)
offset_for_non_ref_pic	0	se(v)
offset_for_top_to_bottom_field	0	se(v)
num_ref_frames_in_pic_order_cnt_cycle	0	ue(v)
for(i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++)		
offset_for_ref_frame[i **]**	0	se(v)
}		
num_ref_frames	0	ue(v)
gaps_in_frame_num_value_allowed_flag	0	u(1)
pic_width_in_mbs_minus1	0	ue(v)
pic_height_in_map_units_minus1	0	ue(v)
frame_mbs_only_flag	0	u(1)
if(!frame_mbs_only_flag)		
mb_adaptive_frame_field_flag	0	u(1)
direct_8x8_inference_flag	0	u(1)
frame_cropping_flag	0	u(1)
if(frame_cropping_flag) {		
frame_crop_left_offset	0	ue(v)
frame_crop_right_offset	0	ue(v)
frame_crop_top_offset	0	ue(v)
frame_crop_bottom_offset	0	ue(v)
}		
vui_parameters_present_flag	0	u(1)

if(vui_parameters_present_flag)		
vui_parameters( )	0	
rbsp_trailing_bits( )	0	
}		

4) Subclause 7.3.2.2 "Picture parameter set RBSP syntax"

Define two new parameters to replace chroma_qp_index_offset when the new profile is indicated, enabling separate control of QP for each of the two chroma components, as follows.

Replace the line of the Picture Parameter Set syntax containing chroma_qp_index_offset with the following:

if ((profile_idc ~~!= 66 && profile_idc != 70 && profile_idc != 71 && profile_idc != 72~~ 11) = = 0)		
chroma_qp_index_offset	1	se(v)
else {		
cb_qp_index_offset	1	se(v)
cr_qp_index_offset	1	se(v)
}		

Replace the corresponding part of the syntax table with the following.

entropy_coding_mode_flag	0	u(1)
if ((profile_idc % 11) != 0) && entropy_coding_mode_flag) {		
transform_8x8_mode_flag	0	u(1)
}		

When chroma_qp_index_offset is not present, chroma_qp_index_offset is equal to cb_qp_index_offset for the Cb array and is equal to chroma_qp_index_offset cr_qp_index_offset for the Cr array. [Ed. Note: Move that statement to the semantics subclause.]

5) Subclause 7.3.5.2 Sub-macroblock prediction syntax

Replace the syntax table with the following.

sub_mb_pred(mb_type) {	C	Descriptor
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
sub_mb_type[mbPartIdx **]**	2	ue(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8)		
if (NumSubMbPart(sub_mb_type[mbPartIdx]) > 1)		
NoMbPartLessThan8x8Flag = 0		
else		
if (!direct_8x8_inference_flag)		
NoMbPartLessThan8x8Flag = 0		
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if((num_ref_idx_l0_active_minus1 > 0 || mb_field_decoding_flag) && mb_type != P_8x8ref0 && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
ref_idx_l0[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
}		

665) Subclause 7.3.5.3 "Residual data syntax"

Replace subclause 7.3.5.3 with the following.

7.3.4.3 Residual data syntax

residual( ) {	C	Descriptor
if(!entropy_coding_mode_flag)		
residual_block = residual_block_cavlc		
else		
residual_block = residual_block_cabac		
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
residual_block(Intra16x16DCLevel, 16)	3	
for(i8x8 = 0; i8x8 < 4; i8x8++) /* each luma 8x8 block */		
for(i4x4 = 0; i4x4 < 4; i4x4++) /* each 4x4 sub-block of block */		
if(CodedBlockPatternLuma & (1 << i8x8)) {		
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
residual_block(Intra16x16ACLevel[i8x8 * 4 + i4x4], 15)	3	
else		
residual_block(LumaLevel[i8x8 * 4 + i4x4], 16)	3 | 4	
} else {		
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
for(i = 0; i < 15; i++)		
Intra16x16ACLevel[i8x8 * 4 + i4x4][i] = 0		
else		
for(i = 0; i < 16; i++)		
LumaLevel[i8x8 * 4 + i4x4][i] = 0		
}		
if(chroma_format_idc = = 0 || chroma_format_idc = = 1) /* Mono, 4:2:0 */		
numC8x8 = 1		
else if(chroma_format_idc = = 2) /* 4:2:2 */		
numC8x8 = 2		
else /* 4:4:4 */		
numC8x8 = 4		
if(CodedBlockPatternChroma & 3) /* chroma DC residual present */		
residual_block(ChromaDCLevel[iCbCr], 4 * numC8x8)	3 | 4	
else		
for(i = 0; i < 4 * numC8x8; i++)		
ChromaDCLevel[iCbCr][i] = 0		
for(iCbCr = 0; iCbCr < 2; iCbCr++)		
for(i8x8 = 0; i8x8 < numC8x8; i8x8++)		
for(i4x4 = 0; i4x4 < 4; i4x4++)		
if(CodedBlockPatternChroma & 2) /* chroma AC residual present */		
residual_block(ChromaACLevel[iCbCr][i8x8*4+i4x4], 15)	3 | 4	
else		
for(i = 0; i < 15; i++)		
ChromaACLevel[iCbCr][i8x8*4+i4x4][i] = 0		
}		

7~~6~~) Subclause 7.3.5 "Macroblock layer syntax"

Replace subclause 7.3.5 with the following.

7.3.5 Macroblock layer syntax

macroblock_layer( ) {	C	Descriptor
mb_type	2	ue(v) | ae(v)
if(mb_type = = I_PCM) {		
while(!byte_aligned())		
pcm_alignment_zero_bit	2	f(1)
for(i = 0; i < 256; i++)		
pcm_sample_luma[i **]**	2	u(v)
for(i = 0; i < 256 * (ChromaFormatFactor – 1); i++)		
pcm_sample_chroma[i **]**	2	u(v)
} else {		
if(MbPartPredMode(mb_type, 0) != Intra_4x4 && MbPartPredMode(mb_type, 0) != Intra_16x16 && NumMbPart(mb_type) = = 4)		
sub_mb_pred(mb_type)	2	
else		
mb_pred(mb_type)	2	
if(MbPartPredMode(mb_type, 0) != Intra_16x16)		
coded_block_pattern	2	me(v) | ae(v)
if(CodedBlockPatternLuma > 0 || CodedBlockPatternChroma > 0 || MbPartPredMode(mb_type, 0) = = Intra_16x16) {		
mb_qp_delta	2	se(v) | ae(v)
residual( )	3 | 4	
}		
}		
}		

8) Subclause 7.3.5 Macroblock layer syntax [Ed. Note: Conflicting edits]

Replace the syntax table with the following.

macroblock_layer( ) {	C	Descriptor
mb_type	2	ue(v) | ae(v)
if(mb_type = = I_PCM) {		
while(!byte_aligned())		
pcm_alignment_zero_bit	2	f(1)
for(i = 0; i < 256 * ChromaFormatFactor; i++)		
pcm_byte[i **]**	2	u(8)
} else {		
NoMbPartLessThan8x8Flag = 1 /* Maybe modified in sub_mb_pred()		
if(MbPartPredMode(mb_type, 0) != Intra_NxN && MbPartPredMode(mb_type, 0) != Intra_16x16 && NumMbPart(mb_type) = = 4) {		
sub_mb_pred(mb_type)	2	
if(NoMbPartLessThan8x8Flag && transform_8x8_mode_flag)		
transform_size_flag	2	ae(v)
} else {		
if(MbPartPredMode(mb_type, 0) != Intra_16x16 && transform_8x8_mode_flag)		
transform_size_flag	2	ae(v)
mb_pred(mb_type)	2	
}		
if(MbPartPredMode(mb_type, 0) != Intra_16x16)		
coded_block_pattern	2	me(v) | ae(v)
if(CodedBlockPatternLuma > 0 || CodedBlockPatternChroma > 0 || MbPartPredMode(mb_type, 0) = = Intra_16x16) {		
mb_qp_delta	2	se(v) | ae(v)
residual( )	3 | 4	
}		
}		
}		

9) Subclause 7.3.5.1 Macroblock prediction syntax

Replace the corresponding part of the syntax table with the following.

if(MbPartPredMode(mb_type, 0) = = Intra_4x4 || MbPartPredMode(mb_type, 0) = = Intra_8x8 || MbPartPredMode(mb_type, 0) = = Intra_16x16) {		
if(MbPartPredMode(mb_type, 0) = = Intra_4x4)		
for(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]**	2	u(1) | ae(v)
if(!prev_intra4x4_pred_mode_flag**[** luma4x4BlkIdx **]**)		
rem_intra4x4_pred_mode[luma4x4BlkIdx **]**	2	u(3) | ae(v)
}		
if(MbPartPredMode(mb_type, 0) = = Intra_8x8)		
for(luma8x8BlkIdx=0; luma8x8BlkIdx<4; luma8x8BlkIdx++) {		
prev_intra8x8_pred_mode_flag[luma8x8BlkIdx **]**	2	ae(v)
if(!prev_intra8x8_pred_mode_flag[luma8x8BlkIdx])		
rem_intra8x8_pred_mode[luma8x8BlkIdx **]**	2	ae(v)
}		

10~~07~~) Subclause 7.3.5.3.1 "Residual block CAVLC syntax"

Substitute the following for the entire content of subclause 7.3.5.3.1 "Residual Block CAVLC Syntax":

residual_block_cavlc(coeffLevel, maxNumCoeff) {	C	Descriptor
for(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
coeff_token	3 | 4	ce(v)
if(TotalCoeff(coeff_token) > 0) {		
if(TotalCoeff(coeff_token) > 10 && TrailingOnes(coeff_token) < 3)		
suffixLength = 1		
else		
suffixLength = 0		
for(i = 0; i < TotalCoeff(coeff_token); i++)		
if(i < TrailingOnes(coeff_token)) {		
trailing_ones_sign_flag	3 | 4	u(1)
level[i] = 1 – 2 * trailing_ones_sign_flag		
} else {		
level_prefix	3 | 4	ce(v)
levelCode = (min(15,level_prefix) << suffixLength)		
if(suffixLength > 0 || level_prefix >= 14) {		
level_suffix	3 | 4	u(v)
levelCode += level_suffix		
}		
if(level_prefix >= 15 && suffixLength == 0)		
levelCode += 15		
if(level_prefix >= 16)		
levelCode += (1<<(level_prefix – 3)) – 4096		
if(i == TrailingOnes(coeff_token) && TrailingOnes(coeff_token) < 3)		
levelCode += 2		
if(levelCode % 2 == 0)		
level[i] = (levelCode + 2) >> 1		
else		
level[i] = (–levelCode – 1) >> 1		
if(suffixLength == 0)		
suffixLength = 1		
if(Abs(level[i]) > (3 << (suffixLength – 1)) && suffixLength < 6)		
suffixLength++		
}		
if(TotalCoeff(coeff_token) < maxNumCoeff) {		
total_zeros	3 | 4	ce(v)
zerosLeft = total_zeros		
} else		
zerosLeft = 0		
for(i = 0; i < TotalCoeff(coeff_token) – 1; i++) {		
if(zerosLeft > 0) {		
run_before	3 | 4	ce(v)
run[i] = run_before		

} else		
run[i] = 0		
zerosLeft = zerosLeft – run[i]		
}		
run[TotalCoeff(coeff_token) – 1] = zerosLeft		
coeffNum = -1		
for(i = TotalCoeff(coeff_token) – 1; i >= 0; i--) {		
coeffNum += run[i] + 1		
coeffLevel[coeffNum] = level[i]		
}		
}		
}		

11) Subclause 7.3.5.4 "Residual block CABAC syntax"

Replace the syntax table with the following.

residual_block_cabac(coeffLevel, maxNumCoeff) {	C	Descriptor
if (maxNumCoeff != 64)		
coded_block_flag /* indicates per 4x4 block whether non-zero coeffs are present */	3 | 4	ae(v)
else		
coded_block_flag = 1 /* for 8x8 blocks, presence of non-zero coeffs is implied */		
if(coded_block_flag) {		
numCoeff = maxNumCoeff		
i = 0		
do {		
significant_coeff_flag[i **]**	3 | 4	ae(v)
if(significant_coeff_flag[i]) {		
last_significant_coeff_flag[i **]**	3 | 4	ae(v)
if(last_significant_coeff_flag[i]) {		
numCoeff = i + 1		
for(j = numCoeff; j < maxNumCoeff; j++)		
coeffLevel[j] = 0		
}		
}		
i++		
} while(i < numCoeff-1)		
coeff_abs_level_minus1[numCoeff-1 **]**	3 | 4	ae(v)
coeff_sign_flag[numCoeff-1 **]**	3 | 4	ae(v)
coeffLevel[numCoeff-1] = (coeff_abs_level_minus1[numCoeff – 1] + 1) * (1 – 2 * coeff_sign_flag[numCoeff – 1])		
for(i = numCoeff-2; i >= 0; i--) {		
if(significant_coeff_flag[i]) {		
coeff_abs_level_minus1[i **]**	3 | 4	ae(v)
coeff_sign_flag[i **]**	3 | 4	ae(v)
coeffLevel[i] = (coeff_abs_level_minus1[i] + 1) * (1 – 2 * coeff_sign_flag[i])		
} else		
coeffLevel[i] = 0		
}		
} else		
for(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
}		

12) Subclause 7.3.5.3 "Residual data syntax"

Replace the corresponding part of the syntax table with the following.

for(i8x8 = 0; i8x8 < 4; i8x8++) /* each luma 8x8 block */		
if(!transform_8x8_mode_flag || !transform_size_flag || !NoMbPartLessThan8x8Flag || MbPartPredMode(mb_type, 0) = = Intra_16x16)		
for(i4x4 = 0; i4x4 < 4; i4x4++) /* each 4x4 sub-block of block */		
if(CodedBlockPatternLuma & (1 << i8x8)) {		
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
residual_block(Intra16x16ACLevel[i8x8 * 4 + i4x4], 15)	3	
else		
residual_block(LumaLevel[i8x8 * 4 + i4x4], 16)	3 | 4	
} else {		
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
for(i = 0; i < 15; i++)		
Intra16x16ACLevel[i8x8 * 4 + i4x4][i] = 0		
else		
for(i = 0; i < 16; i++)		
LumaLevel[i8x8 * 4 + i4x4][i] = 0		
}		
else		
if(CodedBlockPatternLuma & (1 << i8x8))		
residual_block(LumaLevel64[i8x8], 64)	3 | 4	
else		
for(i = 0; i < 64; i++)		
LumaLevel64[i8x8][i] = 0		

13~~13~~8) Subclause 7.4.2.1 "Sequence parameter set RBSP semantics"

[Ed. Note: Add the following where appropriate.]

constraint_set3_flag indicates the following.

– If profile_idc is equal to 55, 66, or 77, the following applies.

  – If level_idc is equal to 11, constraint_set3_flag equal to 1 indicates that the bitstream obeys all constraints specified in Annex A for level 1b and constraint_set3_flag equal to 0 .indicates that the bitstream may or may not obey all constraints specified in Annex A for level 1b.

  – Otherwise (level_idc is not equal to 11), the value of 1 for constraint_set3_flag is reserved for future use by ITU-T | ISO/IEC. constraint_set3_flag shall be equal to 0 in bitstreams conforming to this Recommendation | International Standard. Decoders conforming to this Recommendation | International Standard shall ignore the value of constraint_set3_flag.

– Otherwise (profile_idc is not equal to 55, 66, or 77), constraint_set3_flag equal to 1 indicates that the bitstream obeys all constraints specified in subclauses A.2.4 and A.2.5, and constraint_set3_flag equal to 0 indicates that the bitstream may or may not obey all constraints specified in subclauses A.2.4 and A.2.5. [Ed. Note: Check subclause numbers for 4:2:2/8 and 4:2:0/10 profiles.]

~~equal to 1, when level_idc is equal to 11, indicates that the bitstream obeys all constraints specified in Annex A for level 1b. constraint_set3_flag equal to 0, when level_idc is equal to 11, indicates that the bitstream may or may not obey all constraints specified in Annex A for level 1b.~~

~~When level_idc is not equal to 11, and constraint_set3_flag is equal to 1, and profile_idc is equal to 69 [Ed. Note: check for 4:2:0/10] indicates that the bitstream also satisfies the constraints of subclause A.2.5. [Ed. Check for 4:2:2/8]~~

~~When level_idc is not equal to 11, and constraint_set3_flag is equal to 1, and profile_idc is equal to 70 [Ed. Note: Check for 4:2:2/8] indicates that the bitstream also satisfies the constraints of subclause A.2.4 [Ed. Note: Check for 4:2:0/8].~~

~~When level_idc is not equal to 11, and constraint_set3_flag is equal to 0, the bitstream may or may not satisfy the constraints of subclause A.2.4 and A.2.5.~~

~~constraint_set3_flag shall be equal to 0 in bitstreams conforming to this Recommendation | International Standard. The use of constraint_set3_flag equal to 0 when level_idc is not equal to 11 is reserved, and this combination may be allowed in the future as specified by ITU-T | ISO/IEC. Decoders shall ignore the value of constraint_set3_flag when level_idc is not equal to 11.~~

~~**constraint_set4_flag …**~~

reserved_zero_4~~3~~bits ~~…~~ shall be equal to 0 in bitstreams conforming to this Recommendation | International Standard. Other values of reserved_zero_4bits may be specified in the future by ITU-T | ISO/IEC. Decoders shall ignore the value of reserved_zero_4bits.

-

chroma_format_idc …

lossless_qp0~~coding~~_flag equal to 1 indicates …

bit_depth_luma_minus8 specifies the bit depth of the samples of the luma array and the value of the luma quanti~~s~~zation parameter range offset $QpBdOffset_Y$, as specified by

$$BitDepth_Y = 8 + bit\_depth\_luma\_minus8 \quad (7\text{-}x)$$

$$QpBdOffset_Y = 6 * bit\_depth\_luma\_minus8 \quad (7\text{-}x)$$

When bit_depth_luma_minus8 is not present, the value of bit_depth_luma_minus8 shall be inferred to be equal to 0. bit_depth_luma_minus8 shall be in the range of 0 to 4, inclusive.

bit_depth_chroma_minus8 specifies the bit depth of the samples of the chroma arrays and the value of the chroma quanti~~s~~zation parameter range offset $QpBdOffset_C$, as specified by

$$BitDepth_C = 8 + bit\_depth\_chroma\_minus8 \quad (7\text{-}x)$$

$$QpBdOffset_C = 6 * bit\_depth\_chroma\_minus8 \quad (7\text{-}x)$$

When bit_depth_chroma_minus8 is not present, the value of bit_depth_chroma_minus8 shall be inferred to be equal to 0. bit_depth_chroma_minus8 shall be in the range of 0 to 4, inclusive.

14) Subclause 7.4.2.2 Picture parameter set RBSP semantics

*In the description of pic_init_qp_minus26 in subclause 7.4.2.2, replace the sentence "*The value of pic_init_qp_minus26 shall be in the range of -26 to +25, inclusive*" with "*The value of pic_init_qp_minus26 shall be in the range of (-26 – $QpBdOffset_Y$) to +25, inclusive*".*

Add the following. [Ed. Note: Where?]

transform_8x8_mode_flag equal to 1 indicates that the 8x8 transform decoding process may be in use (see subclause 8.4). transform_8x8_mode_flag equal to 0 indicates that the 8x8 transform decoding process is not in use. If transform_8x8_mode_flag is not present in the bitstream, it shall be inferred to be 0.

~~9) Subclause 7.4.2.2 "Picture parameter set RBSP semantics"~~

~~*In the description of pic_init_qp_minus26 in subclause 7.4.2.2, replace the sentence "*~~~~The value of pic_init_qp_minus26 shall be in the range of -26 to +25, inclusive~~~~" with "~~~~The value of pic_init_qp_minus26 shall be in the range of (-26 - QpBdOffset$_Y$) to +25, inclusive~~~~".~~

15) Subclause 7.4.2.2 Picture parameter set RBSP semantics [Ed. Note: Wrong subclause]

Change the corresponding parts of the subclause with the following.

Add the following paragraph.

transform_size_flag equal to 0 specifies that for the current macroblock the transform coefficient decoding process and picture construction process prior to deblocking filter process for residual 4x4 blocks shall be invoked for luma samples. transform_size_flag equal to 1 specifies that for the current macroblock the transform coefficient decoding process and picture construction process prior to deblocking filter process for residual 8x8 blocks shall be invoked for luma samples. If transform_size_flag is not present in the bitstream, it shall be inferred to be equal to 0.

The variable TransformSizeIs8x8Flag is specified as follows.

– If mb_type of the current macroblock is not equal to Intra_16x16 and transform_size_flag is equal to 1 and transform_8x8_mode_flag is equal to 1, the variable TransformSizeIs8x8Flag is set equal to 1.

– Otherwise, variable TransformSizeIs8x8Flag is set equal to 0.

Change the following parts.

Table 7-8 – Macroblock types for I slices

mb_type	Name of mb_type	transform_size_flag	MbPartPredMode (mb_type, 0)	Intra16x16PredMode	CodedBlockPatternChroma	CodedBlockPatternLuma
0	I_NxN	0	Intra_4x4	na	na	na
0	I_NxN	1	Intra_8x8	na	na	na
1	I_16x16_0_0_0	na	Intra_16x16	0	0	0
2	I_16x16_1_0_0	na	Intra_16x16	1	0	0
3	I_16x16_2_0_0	na	Intra_16x16	2	0	0
4	I_16x16_3_0_0	na	Intra_16x16	3	0	0
5	I_16x16_0_1_0	na	Intra_16x16	0	1	0
6	I_16x16_1_1_0	na	Intra_16x16	1	1	0
7	I_16x16_2_1_0	na	Intra_16x16	2	1	0
8	I_16x16_3_1_0	na	Intra_16x16	3	1	0
9	I_16x16_0_2_0	na	Intra_16x16	0	2	0
10	I_16x16_1_2_0	na	Intra_16x16	1	2	0

11	I_16x16_2_2_0	na	Intra_16x16	2	2	0
12	I_16x16_3_2_0	na	Intra_16x16	3	2	0
13	I_16x16_0_0_1	na	Intra_16x16	0	0	15
14	I_16x16_1_0_1	na	Intra_16x16	1	0	15
15	I_16x16_2_0_1	na	Intra_16x16	2	0	15
16	I_16x16_3_0_1	na	Intra_16x16	3	0	15
17	I_16x16_0_1_1	na	Intra_16x16	0	1	15
18	I_16x16_1_1_1	na	Intra_16x16	1	1	15
19	I_16x16_2_1_1	na	Intra_16x16	2	1	15
20	I_16x16_3_1_1	na	Intra_16x16	3	1	15
21	I_16x16_0_2_1	na	Intra_16x16	0	2	15
22	I_16x16_1_2_1	na	Intra_16x16	1	2	15
23	I_16x16_2_2_1	na	Intra_16x16	2	2	15
24	I_16x16_3_2_1	na	Intra_16x16	3	2	15
25	I_PCM	na	na	na	na	na

~~I_4x4: the macroblock is coded as an Intra_4x4 prediction macroblock~~ I_NxN: A mnemonic name for mb_type equal to 0 with MbPartPredMode(mb_type, 0) equal to Intra_4x4 or Intra_8x8.

Intra_4x4 specifies the macroblock prediction mode and specifies that the Intra_4x4 prediction process is invoked as specified in subclause 8.3.1. Intra_4x4 is an Intra macroblock prediction mode.

Intra_8x8 specifies the macroblock prediction mode and specifies that the Intra_8x8 prediction process is invoked as specified in subclause 8.3.2. Intra_8x8 is an Intra macroblock prediction mode.

161~~610~~) Subclause 7.4.3 "Slice header semantics"

In subclause 7.4.3, after Equation 7-16, replace the phrase "QP_Y is in the range of 0 to 51, inclusive*" with "*SliceQP$_Y$ is in the range of -QpBdOffset$_Y$ to 51, inclusive*".*

171~~711~~) Subclause 7.4.5 "Macroblock layer semantics"

In subclause 7.4.5, make the following changes.

*Replace the paragraph that starts with "***mb_type***" with the following.*

mb_type specifies the macroblock type. The semantics of mb_type depend on the slice type.

Tables and semantics are specified for the various macroblock types for I, SI, P, SP, and B slices. Each table presents the value of mb_type, the name of mb_type, the number of macroblock partitions used (given by the NumMbPart(mb_type) function), the prediction mode of the macroblock (when it is not partitioned) or the first partition (given by the MbPartPredMode(mb_type, 0) function) and the prediction mode of the second partition (given by the MbPartPredMode(mb_type, 1) function). When a value is not applicable it is designated by “na”. In the text, the value

of mb_type may be referred to as the macroblock type and a value X of MbPartPredMode( ) may be referred to in the text by "X macroblock (partition) prediction mode" or as “X prediction macroblocks”.

Table 7-7 shows the allowed collective macroblock types for each slice_type.

NOTE – There are some macroblock types with Pred_L0 prediction mode that are classified as B macroblock types.

Table 7-7 – Allowed collective macroblock types for slice_type

slice_type	allowed collective macroblock types
I (slice)	I (see Table 7-8) (macroblock types)
P (slice)	P (see Table 7-10) and I (see Table 7-8) (macroblock types)
B (slice)	B (see Table 7-11) and I (see Table 7-8) (macroblock types)
SI (slice)	SI (see Table 7-9) and I (see Table 7-8) (macroblock types)
SP (slice)	P (see Table 7-10) and I (see Table 7-8) (macroblock types)

Macroblock types that may be collectively referred to as I macroblock types are specified in Table 7-8.

The macroblock types for I slices are all I macroblock types.

Table 7-8 – Macroblock types for I slices

mb_type	Name of mb_type	MbPartPredMode (mb_type, 0)	Intra16x16PredMode	CodedBlockPatternChroma	CodedBlockPatternLuma
0	I_4x4	Intra_4x4	na	Equation 7-22	Equation 7-22
1	I_16x16_0_0_0	Intra_16x16	0	0	0
2	I_16x16_1_0_0	Intra_16x16	1	0	0
3	I_16x16_2_0_0	Intra_16x16	2	0	0
4	I_16x16_3_0_0	Intra_16x16	3	0	0
5	I_16x16_0_1_0	Intra_16x16	0	1	0
6	I_16x16_1_1_0	Intra_16x16	1	1	0
7	I_16x16_2_1_0	Intra_16x16	2	1	0
8	I_16x16_3_1_0	Intra_16x16	3	1	0
9	I_16x16_0_2_0	Intra_16x16	0	2	0
10	I_16x16_1_2_0	Intra_16x16	1	2	0
11	I_16x16_2_2_0	Intra_16x16	2	2	0
12	I_16x16_3_2_0	Intra_16x16	3	2	0
13	I_16x16_0_0_1	Intra_16x16	0	0	15
14	I_16x16_1_0_1	Intra_16x16	1	0	15
15	I_16x16_2_0_1	Intra_16x16	2	0	15
16	I_16x16_3_0_1	Intra_16x16	3	0	15
17	I_16x16_0_1_1	Intra_16x16	0	1	15
18	I_16x16_1_1_1	Intra_16x16	1	1	15
19	I_16x16_2_1_1	Intra_16x16	2	1	15
20	I_16x16_3_1_1	Intra_16x16	3	1	15
21	I_16x16_0_2_1	Intra_16x16	0	2	15
22	I_16x16_1_2_1	Intra_16x16	1	2	15
23	I_16x16_2_2_1	Intra_16x16	2	2	15
24	I_16x16_3_2_1	Intra_16x16	3	2	15
25	I_PCM	na	na	na	na

The following semantics are assigned to the macroblock types in Table 7-8:

I_4x4: the macroblock is coded as an Intra_4x4 prediction macroblock.

I_16x16_0_0_0, I_16x16_1_0_0, I_16x16_2_0_0, I_16x16_3_0_0, I_16x16_0_1_0, I_16x16_1_1_0, I_16x16_2_1_0, I_16x16_3_1_0, I_16x16_0_2_0, I_16x16_1_2_0, I_16x16_2_2_0, I_16x16_3_2_0, I_16x16_0_0_1, I_16x16_1_0_1, I_16x16_2_0_1, I_16x16_3_0_1, I_16x16_0_1_1, I_16x16_1_1_1, I_16x16_2_1_1, I_16x16_3_1_1, I_16x16_0_2_1, I_16x16_1_2_1, I_16x16_2_2_1, I_16x16_3_2_1: the macroblock is coded as an Intra_16x16 prediction mode macroblock.

To each Intra_16x16 prediction macroblock, an Intra16x16PredMode is assigned, which specifies the Intra_16x16 prediction mode. CodedBlockPatternChroma contains the coded block pattern value for chroma as specified in Table 7-12. When chroma_format_idc is equal to 0, CodedBlockPatternChroma shall be equal to 0. CodedBlockPatternLuma specifies whether, for the luma component, non-zero AC transform coefficient levels are present. CodedBlockPatternLuma equal to 0 specifies that all AC transform coefficient levels in the luma component of the macroblock are equal to 0. CodedBlockPatternLuma equal to 15 specifies that at least one AC transform coefficient levels in the luma component of the macroblock is non-zero, requiring scanning of AC transform coefficient levels for all 16 of the 4x4 blocks in the 16x16 block.

.....

*Replace the paragraph that starts with "***pcm_byte**[i]*" with the following.*

pcm_sample_luma[i] is a sample value. The first pcm_sample_luma[i] values represent luma sample values in the raster scan within the macroblock. The number of bits used to represent each of these samples is $BitDepthLuma_Y$. When profile_idc is not equal to 69, 70, ~~or~~ 71, or 72, pcm_sample_luma[i] shall not be equal to 0.

pcm_sample_chroma[i] is a sample value. The first 128 * (ChromaFormatFactor – 1) pcm_sample_chroma[i] values represent Cb sample values in the raster scan within the macroblock and the remaining 128 * (ChromaFormatFactor – 1) pcm_sample_chroma[i] values represent Cr sample values in the raster scan within the macroblock. The number of bits used to represent each of these samples is $BitDepthLuma_C$. When profile_idc is not equal to 69, 70, ~~or~~ 71, or 72, pcm_sample_chroma[i] shall not be equal to 0.

Replace Equation 7-23 with

$$QP_Y = ((QP_{Y,PREV} + mb\_qp\_delta + 52 + QpBdOffset_Y) \% (52 + QpBdOffset_Y)) - QpBdOffset_Y \quad (7\text{-}23)$$

At the end of the subclause, insert the following text and equation

The value of QP'_Y is derived as

$$QP'_Y = QP_Y + QpBdOffset_Y \quad (7\text{-}23')$$

18) Subclause 7.4.5.1 Macroblock prediction semantics

Add the following paragraph.

prev_intra8x8_pred_mode_flag[luma8x8BlkIdx] and **rem_intra8x8_pred_mode**[luma8x8BlkIdx] specify the Intra_8x8 prediction of the 8x8 luma block with index luma8x8BlkIdx = 0..3.

19) Subclause 7.4.5.2 Sub-macroblock prediction semantics

Add the following paragraph.

The variable NoMbPartLessThan8x8Flag indicates whether for each of the 4 8x8 blocks indexed by mbPartIdx = 0..3 the corresponding SubMbPartWidth(sub_mb_type[mbPartIdx])and SubMbPartHeight(sub_mb_type[mbPartIdx]) are both equal to 8.

20) Subclause 7.4.5.2 Sub-macroblock prediction semantics

Change the following paragraph.

- Depending on MbPartPredMode(mb_type, 0), the following applies.
  - If MbPartPredMode(mb_type, 0) is equal to Intra_16x16, the transform coefficient levels are parsed into the list Intra16x16DCLevel and into the 16 lists Intra16x16ACLevel[i]. Intra16x16DCLevel contains the 16 transform coefficient levels of the DC transform coefficient levels for each 4x4 luma block. For each of the 16 4x4 luma blocks indexed by i = 0..15, the 15 AC transform coefficients levels of the i-th block are parsed into the i-th list Intra16x16ACLevel[i].
  - Otherwise (MbPartPredMode(mb_type, 0) is not equal to Intra_16x16), the following applies.
    - If transform_size_flag is equal to 0 or NoMbPartLessThan8x8Flag is equal to 0 or transform_8x8_mode_flag is equal to 0, for each of the 16 4x4 luma blocks indexed by i = 0..15, the 16 transform coefficient levels of the i-th block are parsed into the i-th list LumaLevel[i].
    - Otherwise (transform_size_flag is equal to 1 and NoMbPartLessThan8x8Flag is equal to 1 and transform_8x8_mode_flag is equal to 1), for each of the 4 8x8 luma blocks indexed by i = 0..3, the 64 transform coefficient levels of the i-th block are parsed into the i-th list LumaLevel64[i].

212~~112~~) Subclause 7.4.5.3 "Residual data semantics"

Replace subclause 7.4.5.3 with the following.

Depending on mb_type, luma or chroma, and chroma format, the syntax structure residual_block(coeffLevel, maxNumCoeff) is used with the arguments coeffLevel, which is a list containing the maxNumCoeff transform coefficient levels that are parsed in residual_block( ), and maxNumCoeff as follows.

- Depending on MbPartPredMode(mb_type, 0), the following applies.
  - If MbPartPredMode(mb_type, 0) is equal to Intra_16x16, the transform coefficient levels are parsed into the list Intra16x16DCLevel and into the 16 lists Intra16x16ACLevel[i]. Intra16x16DCLevel contains the 16 transform coefficient levels of the DC transform coefficient levels for each 4x4 luma block. For each of the 16 4x4 luma blocks indexed by i = 0..15, the 15 AC transform coefficients levels of the i-th block are parsed into the i-th list Intra16x16ACLevel[i].
  - Otherwise (MbPartPredMode(mb_type, 0) is not equal to Intra_16x16), for each of the 16 4x4 luma blocks indexed by i = 0..15, the 16 transform coefficient levels of the i-th block are parsed into the i-th list LumaLevel[i].
- If residue_transform_flag is equal to 0 or MbPartPredMode(mb_type, 0) is not intra_4x4, for each chroma component, indexed by iCbCr = 0..1, the DC transform coefficient levels of the 4x4 chroma blocks are parsed into iCbCr-th list ChromaDCLevel[iCbCr].
- For each of the 4x4 chroma blocks, indexed by i4x4 = 0..3 and i8x8 = 0…num8x8, of each chroma component, indexed by iCbCr = 0..1, the 15 AC transform coefficient levels are parsed into the i4x4-th list of the iCbCr-th chroma component ChromaACLevel[iCbCr][i4x4].

22) Subclause 7.4.5.3.1 Residual block CABAC semantics

Change the following paragraph.

coded_block_flag specifies for blocks that are not 8x8 luma blocks whether the block contains non-zero transform coefficient levels as follows.

23) Subclause 8.3 Intra prediction process

Change the following paragraph.

Inputs to this process are constructed samples prior to the deblocking filter process from neighbouring macroblocks and for Intra_NxN (where NxN is equal to 4x4 or 8x8) prediction mode, the associated values of IntraNxNPredMode from neighbouring macroblocks.

Outputs of this process are specified as follows.

- If mb_type is not equal to I_PCM, the Intra prediction samples of components of the macroblock or in case of the Intra_NxN prediction process for luma samples, the outputs are NxN luma sample arrays as part of the 16x16 luma array of prediction samples of the macroblock.
- Otherwise (mb_type is equal to I_PCM), constructed macroblock samples prior to the deblocking filter process.

Depending on the value of mb_type the following applies.

- If mb_type is equal to I_PCM, the process specified in subclause 8.3.5 is invoked.
- Otherwise (mb_type is not equal to I_PCM), the following applies.
  - The decoding processes for Intra prediction modes are described for the luma component as follows.
  - If the macroblock prediction mode is equal to Intra_4x4, the specification in subclause 8.3.1 applies.
  - Otherwise, if the macroblock prediction mode is equal to Intra_8x8, the specification in subclause 8.3.2 [Ed. Note (TW): new subclause] applies.
  - Otherwise (the macroblock prediction mode is equal to Intra_16x16), the specification in subclause 8.3.3 [Ed. Note (TW): incremented because of new subclause] applies.
  - The decoding processes for Intra prediction modes for the chroma components are described in subclause 8.3.4. [Ed. Note (TW): incremented because of new subclause]

Samples used in the Intra prediction process shall be sample values prior to alteration by any deblocking filter operations.

24) Subclause 8.3.1 Intra_4x4 prediction process for luma samples

Change the following paragraph.

Inputs to this process are constructed luma samples prior to the deblocking filter process from neighbouring macroblocks and the associated values of Intra4x4PredMode or Intra8x8PredMode from the neighbouring macroblocks or macroblock pairs.

25) Subclause 8.3.1.1 Derivation process for the Intra4x4PredMode

Change the following paragraph.

Inputs to this process are the index of the 4x4 luma block luma4x4BlkIdx, the index of the 8x8 luma block luma8x8BlkIdx, and variable arrays Intra4x4PredMode and Intra8x8PredMode that are previously (in decoding order) derived for adjacent macroblocks.

Change the following paragraph.

Let intraMxMPredModeA and intraMxMPredModeB be variables that specify the intra prediction modes of neighbouring 4x4 or 8x8 luma blocks. [Ed. Note (TW): improve clarity]

Intra4x4PredMode[luma4x4BlkIdx] is derived as follows.

- The process specified in subclause 6.4.7.3 is invoked with luma4x4BlkIdx given as input and the output is assigned to mbAddrA, luma4x4BlkIdxA, mbAddrB, and luma4x4BlkIdxB.
- The process specified in subclause 6.4.7.2 is invoked with luma8x8BlkIdx given as input and the output is assigned to mbAddrA, luma8x8BlkIdxA, mbAddrB, and luma8x8BlkIdxB.
- The variable dcOnlyPredictionFlag is derived as follows.
- ...
- For N being either replaced by A or B, the variables intraMxMPredModeN are derived as follows.
  - If dcOnlyPredictionFlag is equal to 1 or the macroblock with address mbAddrN is not coded in Intra_4x4 or Intra_8x8 macroblock prediction mode, intraMxMPredModeN is set equal to 2 (Intra_4x4_DC prediction mode).
  - Otherwise (dcOnlyPredictionFlag is equal to 0 and the macroblock with address mbAddrN is coded in Intra_4x4 or Intra_8x8 macroblock prediction mode), the following applies.
    - If the macroblock with address mbAddrN is coded in Intra_4x4 macroblock mode, intraMxMPredModeN is set equal to Intra4x4PredMode[luma4x4BlkIdxN], where Intra4x4PredMode is the variable array assigned to the macroblock mbAddrN.
    - Otherwise (the macroblock with address mbAddrN is coded in Intra_8x8 macroblock mode), intraMxMPredModeN is set equal to Intra8x8PredMode[luma8x8BlkIdxN], where Intra8x8PredMode is the variable array assigned to the macroblock mbAddrN.
- Intra4x4PredMode[luma4x4BlkIdx] is derived by applying the following procedure.

predIntra4x4PredMode = Min(intraMxMPredModeA, intraMxMPredModeB)

26~~26~~~~13~~) Subclause 8.3.1.2.3 "Specification of Intra_4x4_DC prediction mode" Equation 8-50

*In Equation 8-50 of subclause 8.3.1.2.3, replace "*128*" with "*(1 << ($BitDepth_Y$ – 1))*".*

27) New subclause 8.3.2 "Intra_8x8 prediction process for luma samples"

Add a new subclause 8.3.2 as follows.

This process is invoked when the macroblock prediction mode is equal to Intra_8x8.
Inputs to this process are constructed luma samples prior to the deblocking filter process from neighbouring macroblocks and the associated values of Intra4x4PredMode or Intra8x8PredMode from the neighbouring macroblocks or macroblock pairs.

Outputs of this process are 8x8 luma sample arrays as part of the 16x16 luma array of prediction samples of the macroblock $pred_L$.

The luma component of a macroblock consists of 4 blocks of 8x8 luma samples. These blocks are inverse scanned using the 8x8 luma block inverse scanning process as specified in subclause 6.4.2.

For all the 8x8 luma blocks of the luma component of a macroblock with luma8x8BlkIdx = 0..3, the variable Intra8x8PredMode[luma8x8BlkIdx] is derived as specified in subclause 8.3.2.1.

For the each luma block of 8x8 samples indexed using luma8x8BlkIdx = 0..3,
1. The Intra_8x8 sample prediction process in subclause 8.3.2.2 is invoked with luma8x8BlkIdx and constructed samples prior (in decoding order) to the deblocking filter process from adjacent luma blocks as the input and the output are the Intra_8x8 luma prediction samples $pred8x8_L$[x, y] with x, y = 0..7.

2. The position of the upper-left sample of a 8x8 luma block with index luma8x8BlkIdx inside the current macroblock is derived by invoking the inverse 8x8 luma block scanning process in subclause 6.4.2 with luma8x8BlkIdx as the input and the output being assigned to (xO, yO) and x, y = 0..7.

$pred_L[xO + x, yO + y] = pred8x8_L[x, y]$

3. The transform coefficient decoding process and picture construction process prior to deblocking filter process in subclause 8.5 is invoked with $pred_L$ and luma8x8BlkIdx as the input and the constructed samples for the current 8x8 luma block S'_L as the output.

28) New subclause 8.3.2.1 "Derivation process for the Intra8x8PredMode"

Add a new subclause 8.3.2.1 as follows.

Inputs to this process are the index of the 8x8 luma block luma8x8BlkIdx and variable arrays Intra4x4PredMode and Intra8x8PredMode that are previously (in decoding order) derived for adjacent macroblocks.

Output of this process is the variable Intra8x8PredMode[luma8x8BlkIdx].

Table 8-2 specifies the values for Intra8x8PredMode[luma8x8BlkIdx] and the associated names.

Table 8-2 – Specification of Intra8x8PredMode[luma8x8BlkIdx] and associated names

Intra8x8PredMode[luma8x8BlkIdx]	Name of Intra8x8PredMode[luma8x8BlkIdx]
0	Intra_8x8_Vertical (prediction mode)
1	Intra_8x8_Horizontal (prediction mode)
2	Intra_8x8_DC (prediction mode)
3	Intra_8x8_Diagonal_Down_Left (prediction mode)
4	Intra_8x8_Diagonal_Down_Right (prediction mode)
5	Intra_8x8_Vertical_Right (prediction mode)
6	Intra_8x8_Horizontal_Down (prediction mode)
7	Intra_8x8_Vertical_Left (prediction mode)
8	Intra_8x8_Horizontal_Up (prediction mode)

Let intraMxMPredModeA and intraMxMPredModeB be variables that specify the intra prediction modes of neighbouring 4x4 or 8x8 luma blocks.

Intra8x8PredMode[luma8x8BlkIdx] is derived as follows.

- The process specified in subclause 6.4.7.2 is invoked with luma8x8BlkIdx given as input and the output is assigned to mbAddrA, luma8x8BlkIdxA, mbAddrB, and luma8x8BlkIdxB.
- The variable dcOnlyPredictionFlag is derived as follows.
  - If one of the following conditions is true, dcOnlyPredictionFlag is set equal to 1
    - the macroblock with address mbAddrA is not available
    - the macroblock with address mbAddrB is not available
    - the macroblock with address mbAddrA is available and coded in Inter prediction mode and constrained_intra_pred_flag is equal to 1
    - the macroblock with address mbAddrB is available and coded in Inter prediction mode and constrained_intra_pred_flag is equal to 1
  - Otherwise, dcOnlyPredictionFlag is set equal to 0.
- For N being either replaced by A or B, the variables intraMxMPredModeN are derived as follows.
  - If dcOnlyPredictionFlag is equal to 1 or the macroblock with address mbAddrN is not coded in Intra_4x4 or Intra_8x8 macroblock prediction mode, intraMxMPredModeN is set equal to 2 (Intra_8x8_DC prediction mode).

- Otherwise (dcOnlyPredictionFlag is equal to 0 and the macroblock with address mbAddrN is coded in Intra_4x4 or Intra_8x8 macroblock prediction mode), the following applies.
  - If the macroblock with address mbAddrN is coded in Intra_8x8 macroblock mode, IntraMxMPredModeN is set equal to Intra8x8PredMode[luma8x8BlkIdxN], where Intra8x8PredMode is the variable array assigned to the macroblock mbAddrN.
  - If the macroblock with address mbAddrN is coded in Intra_4x4 macroblock mode, intraMxMPredModeN is derived by the following procedure, where Intra4x4PredMode is the variable array assigned to the macroblock mbAddrN.

```
IntraMxMPredModeA = Intra4x4PredMode [ luma8x8BlkIdxA * 4 + 1 ]
IntraMxMPredModeB = Intra4x4PredMode [ luma8x8BlkIdxB * 4 + 2 ]
```

- Intra8x8PredMode[luma8x8BlkIdx] is derived by applying the following procedure.

```
predIntra8x8PredMode = Min( intraMxMPredModeA, intraMxMPredModeB )
if( prev_intra8x8_pred_mode_flag[ luma8x8BlkIdx ] )
    Intra8x8PredMode[ luma8x8BlkIdx ] = predIntra8x8PredMode
else
    if( rem_intra8x8_pred_mode[ luma8x8BlkIdx ] < predIntra8x8PredMode )
        Intra8x8PredMode[ luma8x8BlkIdx ] = rem_intra8x8_pred_mode[ luma8x8BlkIdx ]
    else
        Intra8x8PredMode[ luma8x8BlkIdx ] = rem_intra8x8_pred_mode[ luma8x8BlkIdx ] + 1
```

29) New subclause 8.3.2.2.1 "Specification of Intra_8x8_Vertical prediction mode"

Add a new subclause 8.3.2.2.1 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 0.

This mode shall be used only when the samples p[x, -1] with x = 0..7 are marked as “available for Intra_8x8 prediction”.

The values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived by

$$pred8x8_L[x, y] = p[x, -1], \text{ with } x, y = 0..7 \qquad \text{(Eq.-No.)}$$

30) New subclause 8.3.2.2 "Intra_8x8 sample prediction"

Add a new subclause 8.3.2.2 as follows.

This process is invoked for each 8x8 luma block of a macroblock with prediction mode equal to Intra_8x8 followed by the transform decoding process and picture construction process prior to deblocking for each 8x8 luma block.

Inputs to this process are the index of the 8x8 luma block with index luma8x8BlkIdx and constructed samples prior (in decoding order) to the deblocking filter process from adjacent luma blocks.

Output of this process are the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 for the 8x8 luma block with index luma8x8BlkIdx.

The position of the upper-left sample of a 8x8 luma block with index luma8x8BlkIdx inside the current macroblock is derived by invoking the inverse 8x8 luma block scanning process in subclause 6.4.2 with luma8x8BlkIdx as the input and the output being assigned to (xO, yO).

The 25 neighbouring samples p[x, y] that are constructed luma samples prior to the deblocking filter process, with x = -1, y = -1..7 and x = 0..15, y = -1, are derived as follows.

– The luma location (xN, yN) is specified by

$$xN = xO + x$$

yN = yO + y

– The derivation process for neighbouring locations in subclause 6.4.7 is invoked for luma locations with (xN, yN) as input and mbAddrN and (xW, yW) as output.
– Each sample p[x, y] with x = -1, y = -1..7 and x = 0..15, y = -1 is derived as follows.
  – If any of the following conditions is true, the sample p[x, y] is marked as "not available for Intra_8x8 prediction"

    - mbAddrN is not available,

    - the macroblock mbAddrN is coded in Inter prediction mode and constrained_intra_pred_flag is equal to 1.
    - x is greater than 7 and luma8x8BlkIdx is equal to 3
  – Otherwise, the sample p[x, y] is marked as "available for Intra_8x8 prediction" and the luma sample at luma location (xW, yW) inside the macroblock mbAddrN is assigned to p[x, y].

When samples p[x, -1], with x = 8..15 are marked as "not available for Intra_8x8 prediction," and the sample p[7, -1] is marked as "available for Intra_8x8 prediction," the sample value of p[7, -1] is substituted for sample values p[x, -1], with x = 7..15 and samples p[x, -1], with x = 7..15 are marked as "available for Intra_8x8 prediction".

NOTE – Each block is assumed to be constructed into a frame prior to decoding of the next block.

Depending on Intra8x8PredMode[luma8x8BlkIdx], one of the Intra_8x8 prediction modes specified in subclauses 8.3.2.2.1 to 8.3.2.2.9 shall be used.

31) New subclause 8.3.2.2.2 "Specification of Intra_8x8_Horizontal prediction mode"

Add a new subclause 8.3.2.2.2 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 1.

This mode shall be used only when the samples p[-1, y], with y = 0..7 are marked as "available for Intra_8x8 prediction".

The values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived by

$$pred8x8_L[x, y] = p[-1, y], \text{ with } x,y = 0..7 \qquad \text{(Eq.-No.)}$$

32) New subclause 8.3.2.2.3 "Specification of Intra_8x8_DC prediction mode"

Add a new subclause 8.3.2.2.3 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 2.

The values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived as follows.

- If all samples p[x, -1], with x = 0..7 and p[-1, y], with y = 0..7 are marked as "available for Intra_8x8 prediction", the values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived by

$$pred8x8_L[x, y] = (p[0, -1] + p[1, -1] + p[2, -1] + p[3, -1] + p[4, -1] + p[5, -1] + p[6, -1] + p[7, -1] + p[-1, 0] + p[-1, 1] + p[-1, 2] + p[-1, 3] + p[-1, 4] + p[-1, 5] + p[-1, 6] + p[-1, 7] + 8) >> 4 \qquad \text{(Eq.-No.)}$$

- Otherwise, if samples p[x, -1], with x = 0..7 are marked as "not available for Intra_8x8 prediction" and p[-1, y], with y = 0..7 are marked as "available for Intra_8x8 prediction", the values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived by

$$pred8x8_L[x, y] = (p[-1, 0] + p[-1, 1] + p[-1, 2] + p[-1, 3] + p[-1, 4] + p[-1, 5] + p[-1, 6] + p[-1, 7] + 4) >> 3 \qquad \text{(Eq.-No.)}$$

- Otherwise, if samples p[-1, y], with y = 0..7 are marked as “not available for Intra_8x8 prediction” and p[x, -1], with x = 0 .. 7 are marked as “available for Intra_8x8 prediction”, the values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0 .. 7 are derived by

$$pred8x8_L[x, y] = (p[0, -1] + p[1, -1] + p[2, -1] + p[3, -1] + p[4, -1] + p[5, -1] + p[6, -1] + p[7, -1] + 4) >> 3 \quad \text{(Eq.-No.)}$$

- Otherwise (all samples p[x, -1], with x = 0..7 and p[-1, y], with y = 0..7 are marked as “not available for Intra_8x8 prediction”), the values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived by

$$pred8x8_L[x, y] = 128 \quad \text{(Eq.-No.)}$$

NOTE – An 8x8 luma block can always be predicted using this mode.

33) New subclause 8.3.2.2.4 "Specification of Intra_8x8_Diagonal_Down_Left prediction mode"

Add a new subclause 8.3.2.2.4 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 3.

This mode shall be used only when the samples p[x, -1] with x = 0..15 are marked as “available for Intra_8x8 prediction”.

The values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived as follows.

- If x is equal to 7 and y is equal to 7,

$$pred8x8_L[x, y] = (p[14, -1] + 3 * p[15, -1] + 2) >> 2 \quad \text{(Eq.-No.)}$$

- Otherwise (x is not equal to 7 or y is not equal to 7),

$$pred8x8_L[x, y] = (p[x + y, -1] + 2 * p[x + y + 1, -1] + p[x + y + 2, -1] + 2) >> 2 \quad \text{(Eq.-No.)}$$

34) New subclause 8.3.2.2.5 "Specification of Intra_8x8_Down_Right prediction mode"

Add a new subclause 8.3.2.2.5 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 4.

This mode shall be used only when the samples p[x, -1] with x = 0..7 and p[-1, y] with y = -1..7 are marked as “available for Intra_8x8 prediction”.

The values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived as follows.

- If x is greater than y,

$$pred8x8_L[x, y] = (p[x - y - 2, -1] + 2 * p[x - y - 1, -1] + p[x - y, -1] + 2) >> 2 \quad \text{(Eq.-No.)}$$

- Otherwise if x is less than y,

$$pred8x8_L[x, y] = (p[-1, y - x - 2] + 2 * p[-1, y - x - 1] + p[-1, y - x] + 2) >> 2 \quad \text{(Eq.-No.)}$$

- Otherwise (x is equal to y),

$$pred8x8_L[x, y] = (p[0, -1] + 2 * p[-1, -1] + p[-1, 0] + 2) >> 2 \quad \text{(Eq.-No.)}$$

35) New subclause 8.3.2.2.6 "Specification of Intra_8x8_Vertical_Right prediction mode"

Add a new subclause 8.3.2.2.6 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 5.

This mode shall be used only when the samples p[x, -1] with x = 0..7 and p[-1, y] with y = -1..7 are marked as "available for Intra_8x8 prediction".

Let the variable zVR be set equal to 2 * x – y.

The values of the prediction samples $pred8x8_L$[x, y], with x, y = 0..7 are derived as follows.

- If zVR is equal to 0, 2, 4, 6, 8, 10, 12, or 14

  $pred8x8_L[x, y] = (p[x - (y >> 1) - 1, -1] + p[x - (y >> 1), -1] + 1) >> 1$ (Eq.-No.)

- Otherwise, if zVR is equal to 1, 3, 5, 7, 9, 11, or 13

  $pred8x8_L[x, y] = (p[x - (y >> 1) - 2, -1] + 2 * p[x - (y >> 1) - 1, -1] + p[x - (y >> 1), -1] + 2) >> 2$ (Eq.-No.)

- Otherwise, if zVR is equal to -1,

  $pred8x8_L[x, y] = (p[-1, 0] + 2 * p[-1, -1] + p[0, -1] + 2) >> 2$ (Eq.-No.)

- Otherwise (zVR is equal to -2, -3, -4, -5, -6, or -7),

  $pred8x8_L[x, y] = (p[-1, y - 2*x - 1] + 2 * p[-1, y - 2*x - 2] + p[-1, y - 2*x - 3] + 2) >> 2$ (Eq.-No.)

36) New subclause 8.3.2.2.7 "Specification of Intra_8x8_Horizontal_Down prediction mode"

Add a new subclause 8.3.2.2.7 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 6.

This mode shall be used only when the samples p[x, -1] with x = 0..7 and p[-1, y] with y = -1..7 are marked as "available for Intra_8x8 prediction".

Let the variable zHD be set equal to 2 * y – x.

The values of the prediction samples $pred8x8_L$[x, y], with x, y = 0..7 are derived as follows.

- If zHD is equal to 0, 2, 4, 6, 8, 10, 12, or 14

  $pred8x8_L[x, y] = (p[-1, y -(x >> 1) - 1] + p[-1, y - (x >> 1)] + 1) >> 1$ (Eq.-No.)

- Otherwise, if zHD is equal to 1, 3, 5, 7, 9, 11, or 13

  $pred8x8_L[x, y] = (p[-1, y - (x >> 1) - 2] + 2 * p[-1, y - (x >> 1) - 1] + p[-1, y - (x >> 1)] + 2) >> 2$ (Eq.-No.)

- Otherwise, if zHD is equal to -1,

  $pred8x8_L[x, y] = (p[-1, 0] + 2 * p[-1, -1] + p[0, -1] + 2) >> 2$ (Eq.-No.)

- Otherwise (zHD is equal to –2, -3, -4, -5, -6, -7),

  $pred8x8_L[x, y] = (p[x - 2*y - 1, -1] + 2 * p[x - 2*y - 2, -1] + p[x - 2*y - 3, -1] + 2) >> 2$ (Eq.-No.)

37) New subclause 8.3.2.2.8 "Specification of Intra_8x8_Horizontal_Down prediction mode"

Add a new subclause 8.3.2.2.8 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 7.

This mode shall be used only when the samples p[x, -1] with x = 0..15 are marked as "available for Intra_8x8 prediction".

The values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived as follows.

- If y is equal to 0, 2, 4 or 6

$$pred8x8_L[x, y] = (p[x + (y >> 1), -1] + p[x + (y >> 1) + 1, -1] + 1) >> 1 \quad \text{(Eq.-No.)}$$

- Otherwise (y is equal to 1, 3, 5, 7),

$$pred8x8_L[x, y] = (p[x + (y >> 1), -1] + 2 * p[x + (y >> 1) + 1, -1] + p[x + (y >> 1) + 2, -1] + 2) >>2 \quad \text{(Eq.-No.)}$$

38) New subclause 8.3.2.2.9 "Specification of Intra_8x8_Horizontal_Down prediction mode"

Add a new subclause 8.3.2.2.9 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 8.

This mode shall be used only when the samples p[-1, y] with y = 0..7 are marked as "available for Intra_8x8 prediction".

Let the variable zHU be set equal to x + 2 * y.

The values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived as follows:

- If zHU is equal to 0, 2, 4, 6, 8, 10, or 12

$$pred8x8_L[x, y] = (p[-1, y + (x >> 1)] + p[-1, y + (x >> 1) + 1] + 1) >> 1 \quad \text{(Eq.-No.)}$$

- Otherwise, if zHU is equal to 1, 3, 5, 7, 9, or 11

$$pred8x8_L[x, y] = (p[-1, y + (x >> 1)] + 2 * p[-1, y + (x >> 1) + 1] + p[-1, y + (x >> 1) + 2] + 2) >>2 \quad \text{(Eq.-No.)}$$

- Otherwise, if zHU is equal to 13,

$$pred8x8_L[x, y] = (p[-1, 6] + 3 * p[-1, 7] + 2) >> 2 \quad \text{(Eq.-No.)}$$

- Otherwise (zHU is greater than 13),

$$pred8x8_L[x, y] = p[-1, 7] \quad \text{(Eq.-No.)}$$

393914) Subclause 8.3.2.3 "Specification of Intra_16x16_DC prediction mode" Equation 8-75

*In Equation 8-75, replace "*128*" with "*$(1 << (BitDepth_Y - 1))$*".*

404015) Subclause 8.3.2.4 "Specification of Intra_16x16_Plane prediction mode" Equation 8-76

*In Equation 8-76, replace "*Clip1*" with "*Clip1_Y*".*

414116) Subclause 8.3.3.1 "Specification of Intra_Chroma_DC prediction mode"

*Rename subclause 8.3.3.1 to "*Specification of Intra_Chroma_DC prediction mode for 4:2:0 chroma format*".*

*In Equations 8-85, 8-88, 8-91, and 8-95, replace "*128*" with "*$(1 << (\text{BitDepth}_\text{C} - 1))$*".*

424217) Subclause 8.3.3.2 "Specification of Intra_Chroma_Horizontal prediction mode"

*Rename subclause 8.3.3.2 to "*Specification of Intra_Chroma_Horizontal prediction mode for 4:2:0 chroma format*".*

434318) Subclause 8.3.3.3 "Specification of Intra_Chroma_Vertical prediction mode"

*Rename subclause 8.3.3.3 to "*Specification of Intra_Chroma_Vertical prediction mode for 4:2:0 chroma format*".*

444419) Subclause 8.3.3.4 “Specification of Intra_Chroma_Plane prediction mode”

*Rename subclause 8.3.3.4 to "*Specification of Intra_Chroma_Plane prediction mode for 4:2:0 chroma format*".*

*In Equation 8-98, replace "*Clip1*" with "*Clip1_C*".*

454520) New subclause 8.3.3.5 "Specification of Intra_Chroma_DC prediction mode for 4:2:2 chroma format" [Ed. Note: Link errors]

Insert a new subclause 8.3.3.5 as follows.

8.3.3.5 Specification of Intra_Chroma_DC prediction mode for 4:2:2 chroma format

[Ed. Note: Possibly merge this content into subclause 8.3.3.1.]

The values of the prediction samples predC[x, y] with x = 0..7 and y = 0..15 are derived as follows.

- If the samples p[x, –1] with x = 0..7 and the samples p[–1, y] and y = 0..15 are marked as "available for Intra chroma prediction",

  pred_C[x, y] = **Error! Objects cannot be created from editing field codes.** ,with x = 0..7 and y = 0..15

- Otherwise, if the samples p[x, –1] with x = 0..7 are marked as "available for Intra chroma prediction" and the samples p[–1, y] with y = 0..15 are marked as "not available for Intra chroma prediction",

  pred_C[x, y] =**Error! Objects cannot be created from editing field codes.**, with x = 0..7 and y = 0..15

- Otherwise, if the samples p[x, –1] with x = 0..7 are marked as "not available for Intra chroma prediction" and the samples p[-1, y] with y = 0..15 are marked as "available for Intra chroma prediction",

  pred_C[x, y] =**Error! Objects cannot be created from editing field codes.**, with x = 0..7 and y = 0..15

- Otherwise (the samples p[x, –1] with x = 0..7 and the samples p[-1, y] with y = 0..15 are marked as "not available for Intra chroma prediction"),

  $pred_C[x, y] = (1 << (BitDepth – 1))$, with x = 0..7 and y = 0..15

464621) New subclause 8.3.3.6 "Specification of Intra_Chroma_Horizontal prediction mode for 4:2:2 chroma format"

Insert a new subclause 8.3.3.6 as follows.

8.3.3.6 Specification of Intra_Chroma_Horizontal prediction mode for 4:2:2 chroma format

[Ed. Note: Possibly merge this content into subclause 8.3.3.2.]

This mode shall be used only when the samples p[-1, y] with y = 0..15 are marked as "available for Intra chroma prediction".

The values of the prediction samples predC[x, y] are derived as follows.

$pred_C[x, y] = p[-1, y]$, with x = 0..7 and y = 0..15

474722) New subclause 8.3.3.7 "Specification of Intra_Chroma_Vertical prediction mode for 4:2:2 chroma format"

Insert a new subclause 8.3.3.7 as follows.

8.3.3.7 Specification of Intra_Chroma_Vertical prediction mode for 4:2:2 chroma format

[Ed. Note: Possibly merge this content into subclause 8.3.3.3.]

This mode shall be used only when the samples p[x, -1] with x = 0..7 are marked as "available for Intra chroma prediction".

The values of the prediction samples predC[x, y] are derived as follows.

$pred_C[x, y] = p[x, -1]$, with x = 0..7 and y = 0..15

484823) New subclause 8.3.3.8 "Specification of Intra_Chroma_Plane prediction mode for 4:2:2 chroma format"

Insert a new subclause 8.3.3.8 as follows.

8.3.3.8 Specification of Intra_Chroma_Plane prediction mode for 4:2:2 chroma format

[Ed. Note: Possibly merge this content into subclause 8.3.3.4.]

This mode shall be used only when the samples p[x, -1], with x = 0..7 and p[-1, y], with y = -1..15 are marked as "available for Intra chroma prediction".

The values of the prediction samples predC[x, y] are derived as follows.

$pred_C[x, y] = Clip1((a + b * (x – 3) + c * (y – 7) + 16) >> 5)$, with x = 0..7 and y = 0..15

where:

$a = 16 * (p[-1, 15] + p[7, -1])$

b = (17 * H + 16) >> 5

c = (5 * V + 32) >> 6

and H and V are specified as follows.

$$H = \sum_{x'=0}^{3} (x'+1) * (p[4+x',-1] - p[2-x',-1])$$

$$V = \sum_{y'=0}^{7} (y'+1) * (p[-1,8+y'] - p[-1,6-y'])$$

494924) New subclause 8.3.3.9 "Specification of intra chroma prediction for 4:4:4 chroma format"

Insert a new subclause 8.3.3.9 as follows.

8.3.3.9 Specification of intra chroma prediction for 4:4:4 chroma format

[Ed. Note: Find correct subclause structure for this once Proposal A or B below has been selected.]

Two alternative methods are under consideration for 4:4:4 intra chroma prediction. These are referred to herein as proposal "A" and proposal "B".

Proposal "A" is specified as follows.

Both Intra_4x4 prediction and Intra_16x16 prediction are applied equally on all color components. Both shall not be used at the same time for each component, and each component shall not use different intra prediction mode. For example, when Intra_4x4 prediction is used, all color components shall use only one mode of Intra_4x4 prediction. (So no separate syntax elements are sent to control chroma intra prediction.)

Proposal "B" is specified as follows.

As is the case currently for 4:2:0 operation, four intra chroma prediction modes are defined and the selection between these is controlled by a syntax element. The four prediction modes are DC, vertical, horizontal, and plane. Due to the increased in the size of the chroma blocks controlled by this prediction operation to a 16x16 block size, the operation of these four modes for chroma prediction is defined in the same manner currently used for 16x16 luma intra prediction.

505025) Subclause 8.3.4 "Sample construction process for I_PCM macroblocks" Equations 8-104 and 8-105

In subclause 8.3.4, replace Equations 8-104 and 8-105 with the following.

```
for( i = 0; i < 256; i++ )
    S'L[ xP + ( i % 16 ), yP + dy * ( i / 16 ) ) ] = pcm_sample_luma[ i ]                    (8-104)

for( i = 0; i < 128 * ( ChromaFormatFactor – 1 ); i++ ) {
    S'Cb[ ( xP >> 1 ) + ( i % 8 ), ( ( yP + 1 ) >> 1 ) + dy * ( i / 8 ) ] = pcm_sample_chroma[ i ]   (8-105)
    S'Cr[ ( xP >> 1 ) + ( i % 8 ), ( ( yP + 1 ) >> 1 ) + dy * ( i / 8 ) ]
                            = pcm_sample_chroma[ i + 128 * ( ChromaFormatFactor – 1) ]
}
```

515126) Subclause 8.4 "Inter prediction process"

In subclause 8.4, make the following changes.

Define variables partWidthC and partHeightC as below.

- If chroma_format_idc is equal to 1,

  partWidthC = partWidth/2 (8-x)

  partHeightC = partHeight/2 (8-x)

- Otherwise, if chroma_format_idc is equal to 2,

  partWidthC = partWidth/2 (8-x)

  partHeightC = partHeight (8-x)

- Otherwise, chroma_format_idc is equal to 3,

  partWidthC = partWidth (8-x)

  partHeightC = partHeight (8-x)

Replace "partWidth/2"and "partHeight/2" to "partWidthC" and "partHeightC", respectively.

Replace "$pred_C$[xP / 2 + xS / 2 + x, yP / 2 + yS / 2 + y]" to as below.

- If chroma_format_idc is equal to 1,

  $pred_C$[xP / 2 + xS / 2 + x, yP / 2 + yS / 2 + y] = $predPart_C$[x, y] (8-x)

- Otherwise, if chroma_format_idc is equal to 2,

  $pred_C$[xP / 2 + xS / 2 + x, yP + yS + y] = $predPart_C$[x, y] (8-x)

- Otherwise, chroma_format_idc is equal to 3,

  $pred_C$[xP + xS + x, yP + yS + y] = $predPart_C$[x, y] (8-x)

525227) Subclause 8.4.1.4 "Derivation process for chroma motion vectors"

In subclause 8.4.1.4, make the following changes.

Add chroma format to input to this process.

Replace "one-eighth sample mvCLX units" to "one-quarter sample mvCLX units in case of chroma_format_idc equal to 3, and one-eighth sample mvCLX units in case of chroma_format_idc equal to 1".

535328) Subclause 8.4.2 "Decoding process for Inter prediction samples"

Replace "partWidth/2"and "partHeight/2" to "partWidthC" and "partHeightC", respectively.

545429) Subclause 8.4.2.2 "Fractional sample interpolation process"

Replace subclause 8.4.2.2, with the following.

Inputs to this process are

- the chroma format,
- the current partition given by its partition index mbPartIdx and its sub-macroblock partition index subMbPartIdx,
- the width and height partWidth, partHeight of this partition in luma-sample units,
- a luma motion vector mvLX given in quarter-luma-sample units,
- a chroma motion vector mvCLX given in eighth-chroma-sample units, and
- the selected reference picture sample arrays $refPicLX_L$, $refPicLX_{Cb}$, and $refPicLX_{Cb}$

Outputs of this process are

- a (partWidth)x(partHeight) array $predPartLX_L$ of prediction luma sample values and
- two (partWidthC)x(partHeightC) arrays $predPartLX_{Cb}$, and $predPartLX_{Cr}$ of prediction chroma sample values.

Let (xA_L, yA_L) be the location given in full-sample units of the upper-left luma sample of the current partition given by mbPartIdx\subMbPartIdx relative to the upper-left luma sample location of the given two-dimensional array of luma samples.

Let ($xInt_L$, $yInt_L$) be a luma location given in full-sample units and ($xFrac_L$, $yFrac_L$) be an offset given in quarter-sample units. These variables are used only inside this subclause for specifying general fractional-sample locations inside the reference sample arrays $refPicLX_L$, $refPicLX_{Cb}$, and $refPicLX_{Cr}$.

For each luma sample location ($0 <= x_L < partWidth$, $0 <= y_L < partHeight$) inside the prediction luma sample array $predLX_L$, the corresponding predicted luma sample value $predLX_L[x_L, y_L]$ is derived as follows:

$$xInt_L = xA_L + (mvLX[0] >> 2) + x_L \quad (8\text{-}176)$$
$$yInt_L = yA_L + (mvLX[1] >> 2) + y_L \quad (8\text{-}177)$$

$$xFrac_L = mvLX[0] \& 3 \quad (8\text{-}178)$$
$$yFrac_L = mvLX[1] \& 3 \quad (8\text{-}179)$$

- The prediction sample value $predLX_L[x_L, y_L]$ is derived by invoking the process specified in subclause 8.4.2.2.1 with ($xInt_L$, $yInt_L$), ($xFrac_L$, $yFrac_L$) and $refPicLX_L$ given as input.

Let ($xInt_C$, $yInt_C$) be a chroma location given in full-sample units and ($xFrac_C$, $yFrac_C$) be an offset given in one-eighth sample units. These variables are used only inside this subclause for specifying general fractional-sample locations inside the reference sample arrays $refPicLX_{Cb}$, and $refPicLX_{Cr}$.

For each chroma sample location ($0 <= x_C < partWidthC$, $0 <= y_C < partHeightC$) inside the prediction chroma sample arrays $predPartLX_{Cb}$ and $predPartLX_{Cr}$, the corresponding prediction chroma sample values $predPartLX_{Cb}[x_C, y_C]$ and $predPartLX_{Cr}[x_C, y_C]$ are derived as follows:

- If chroma_format_idc is equal to 1,

$$xInt_C = (xA_L >> 1) + (mvCLX[0] >> 3) + x_C \quad (8\text{-}180)$$
$$yInt_C = (yA_L >> 1) + (mvCLX[1] >> 3) + y_C \quad (8\text{-}181)$$

$$xFrac_C = mvCLX[0] \& 7 \quad (8\text{-}182)$$
$$yFrac_C = mvCLX[1] \& 7 \quad (8\text{-}183)$$

- If chroma_format_idc is equal to 2,

$$xInt_C = (xA_L >> 1) + (mvCLX[0] >> 3) + x_C \quad (8\text{-}180)$$
$$yInt_C = yA_L + (mvCLX[1] >> 2) + y_C \quad (8\text{-}181)$$

$xFrac_C = mvCLX[0] \& 7$ (8-182)
$yFrac_C = (mvCLX[1] \& 3) << 1$ (8-183)

- If chroma_format_idc is equal to 3,

$xInt_C = xA_L + (mvCLX[0] >> 2) + x_C$ (8-180)
$yInt_C = yA_L + (mvCLX[1] >> 2) + y_C$ (8-181)

$xFrac_C = (mvCLX[0] \& 3) << 1$ (8-182)
$yFrac_C = (mvCLX[1] \& 3) << 1$ (8-183)

- The prediction sample value $predPartLX_{Cb}[x_C, y_C]$ is derived by invoking the process specified in subclause 8.4.2.2.2 with ($xInt_C$, $yInt_C$), ($xFrac_C$, $yFrac_C$) and $refPicLX_{Cb}$ given as input.
- The prediction sample value $predPartLX_{Cr}[x_C, y_C]$ is derived by invoking the process specified in subclause 8.4.2.2.2 with ($xInt_C$, $yInt_C$), ($xFrac_C$, $yFrac_C$) and $refPicLX_{Cr}$ given as input.

555530) Subclause 8.4.2.2.2 “Chroma sample interpolation process”

In subclause 8.4.2.2.2, make the following changes.

[Ed.Note: SUZ-wskim]

565631) Subclause 8.4.2.2.1 “Luma sample interpolation process” Equations 8-187, 8-188, 8-191, 8-192, 8-193

*In Equations 8-187, 8-188, 8-191, 8-192, and 8-193, replace "*Clip1" with "$Clip1_Y$*".*

575732) Subclause 8.4.2.3 “Weighted sample prediction process”

Replace “partWidth/2”and “partHeight/2” to “partWidthC” and “partHeightC”, respectively.

585833) Subclause 8.4.2.3.2 “Weighted sample prediction process” Equations 8-218, 8-219, 8-220, 8-235, 8-236, 8-240, and 8-241

*In Equations 8-218, 8-219, and 8-220, replace "*Clip1*" with "*$Clip1_C$*".*

Multiply the values of o_0 and o_1 in Equations 8-235 and 8-236 by (1 << ($BitDepth_Y$ -8)) and in Equations 8-240 and 8-241 by (1 << ($BitDepth_C$ -8)).

59) Subclause 8.5 Transform coefficient decoding process and picture construction process prior to deblocking filter process

Change subclause 8.5 as follows.

Inputs to this process are Intra16x16DCLevel (if available), Intra16x16ACLevel (if available), LumaLevel (if available), LumaLevel64 (if available), ChromaDCLevel, ChromaACLevel, and available Inter or Intra prediction sample arrays for the current macroblock for the applicable component $pred_L$, $pred_{Cb}$, or $pred_{Cr}$.

NOTE – When decoding a macroblock in Intra_4x4 (or Intra_8x8) prediction mode, the luma component of the macroblock prediction array may not be complete, since for each 4x4 (or 8x8) luma block, the Intra_4x4 (or 8x8) prediction process for luma samples as specified in subclause 8.3.1 (or 8.3.2) and the process specified in this subclause are iterated.

NOTE – When decoding a macroblock in Intra_4x4 (or Intra_8x8) prediction mode, the luma component of the macroblock constructed sample arrays prior to the deblocking filter process may not be complete, since for each 4x4 (or 8x8) luma block, the Intra_4x4 (or 8x8) prediction process for luma samples as specified in subclause 8.3.1 (or 8.3.2) and the process specified in this subclause are iterated.

When the current macroblock is coded as P_Skip or B_Skip, all values of LumaLevel, LumaLevel64, ChromaDCLevel, ChromaACLevel are set equal to 0 for the current macroblock.

6060~~34~~) Subclause 8.5.1 “Specification of transform decoding process for residual blocks” Equation 8-243

In Equation 8-243, replace "Clip1" with "$Clip1_Y$".

61) Subclause 8.5.1 Specification of transform decoding process for 4x4 luma residual blocks [Ed. Note: Conflicting Edits]

Change subclause 8.5.1 as follows.

Change the title as follows.

"Specification of transform decoding process for 4x4 luma residual blocks"

Add the following paragraph at the start.

This process is invoked for luma 4x4 blocks when TransformSizeIs8x8Flag is equal to 0 or for chroma 4x4 blocks.

626~~235~~) Subclause 8.5.2 “Specification of transform decoding process for luma samples of Intra_16x16 macroblock prediction mode” Equation 8-245

In Equation 8-245, replace "Clip1" with "$Clip1_Y$".

63) New subclause 8.5.2 "Specification of transform decoding process for 8x8 luma residual blocks" [Ed. Note: Error in new subclause number]

Add a new subclause 8.5.2 as follows.

This process is invoked when TransformSizeIs8x8Flag is equal to 1. The variable LumaLevel64[luma8x8BlkIdx] with luma8x8BlkIdx = 0..3 contains the levels for the luma transform coefficients for the luma 8x8 block with index luma8x8BlkIdx.

For an 8x8 luma block indexed by luma8x8BlkIdx = 0..3, the following ordered steps are specified.

1. The inverse transform coefficient scanning process as described in subclause 8.5.5 is invoked with LumaLevel64[luma8x8BlkIdx] as the input and the two-dimensional array c as the output.
2. The scaling and transformation process for residual 8x8 blocks as specified in subclause 8.5.6 is invoked with c as the input and r as the output.
3. The position of the upper-left sample of an 8x8 luma block with index luma8x8BlkIdx inside the macroblock is derived by invoking the inverse 8x8 luma block scanning process in subclause 6.4.2.2 with luma8x8BlkIdx as the input and the output being assigned to (xO, yO).

4. The 8x8 array u with elements u_{ij} for i, j = 0..7 is derived as

$$u_{ij} = Clip1(pred_L[xO + j, yO + i] + r_{ij}) \quad (8\text{-}243)$$

5. The picture construction process prior to deblocking filter process in subclause 8.5.9 is invoked with luma8x8BlkIdx, u as the input and S' as the output.

646436) Subclause 8.5.3 "Specification of transform decoding process for chroma samples" Equation 8-250

In Equation 8-250, replace "Clip1" with "$Clip1_C$".

65) Subclause 8.5.4 Inverse scanning process for transform coefficients

Change the title as follows.

"Inverse scanning process for 4x4 transform coefficients"

666637) Subclause 8.5.5 "Derivation process for the quantisation parameters and scaling function"

Replace the sentence "QP quantisation parameter values QP_Y, QP_C, QS_Y, and QS_C shall be in the range of 0 to 51, inclusive" with "QP quantisation parameter values QP_Y and QS_Y shall be in the range of $-QpBdOffset_Y$ to 51, inclusive. QP quantisation parameter values QP_C and QS_C shall be in the range of $-QpBdOffset_C$ to 51, inclusive."

Replace Equation 8-251 with

"$qP_I = Clip3(-QpBdOffset_C, 51, QP_Y + chroma\_qp\_index\_offset)$ (8-251)"

After Table 8-13, insert the following text and equation:

" The value of QP'_C shall be derived as follows.

"$QP'_C = QP_C + QpBdOffset_C$ (8-251')"

67) Subclause 8.5.5 "Derivation process for the quantisation parameters and scaling function" [Ed. Note: Conflicting Edits]

Add to subclause 8.5.5 the following.

The function LevelScale64(m, i, j) is specified as follows:

$$\text{LevelScale64}(m,i,j) = \begin{cases} V_{m0} & \text{for } (i,j) \text{ with } i \in \{0,4\},\ j \in \{0,4\}, \\ V_{m1} & \text{for } (i,j) \text{ with } i \in \{1,3,5,7\},\ j \in \{1,3,5,7\}, \\ V_{m2} & \text{for } (i,j) \text{ with } i \in \{2,6\},\ j \in \{2,6\}, \\ V_{m3} & \text{for } (i,j) \text{ with } (i \in \{0,4\},\ j \in \{1,3,5,7\}) \cup (i \in \{1,3,5,7\},\ j \in \{0,4\}), \\ V_{m4} & \text{for } (i,j) \text{ with } (i \in \{0,4\},\ j \in \{2,6\}) \cup (i \in \{2,6\},\ j \in \{0,4\}), \\ V_{m5} & \text{otherwise;} \end{cases}$$

(Eq.-No.)

where the first and second subscripts of V are row and column indices, respectively, of the matrix specified as:

$$V = \begin{bmatrix} 20 & 18 & 32 & 19 & 25 & 24 \\ 22 & 19 & 35 & 21 & 28 & 26 \\ 26 & 23 & 42 & 24 & 33 & 31 \\ 28 & 25 & 45 & 26 & 35 & 33 \\ 32 & 28 & 51 & 30 & 40 & 38 \\ 36 & 32 & 58 & 34 & 46 & 43 \end{bmatrix}$$

(Eq.-No.)

68) New subclause 8.5.5 "Inverse scanning process for 8x8 luma transform coefficients" [Ed. Note: Bad subclause number]

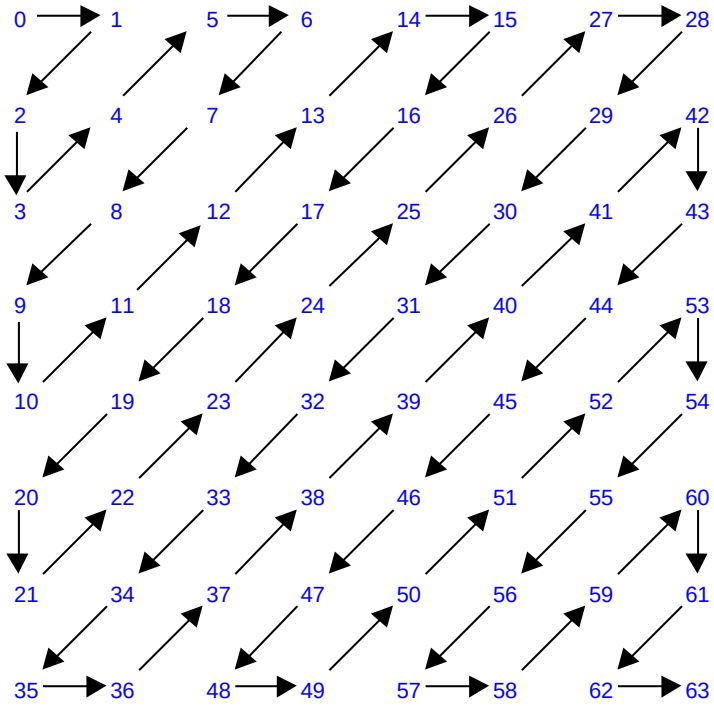
Add a new subclause 8.5.5 as follows.

Input to this process is a list of 64 values.

Output of this process is a variable c containing a two-dimensional array of 8x8 values with level assigned to locations in the transform block.

The decoding process maps the sequence of transform coefficient levels to the transform coefficient level positions. For this mapping, the two inverse scanning patterns shown in Figure 8-9 are used.

The inverse zig-zag scan shall be used for frame macroblocks and the inverse field scan shall be used for field macroblocks.



a) Zig-zag scan

0	3	8	15	22	30	38	52
1	4	14	21	29	37	45	53
2	7	16	23	31	39	46	58
5	9	20	28	36	44	51	59
6	13	24	32	40	47	54	60
10	17	25	33	41	48	55	61
11	18	26	34	42	49	56	62
12	19	27	35	43	50	57	63

b) Field scan [Ed. Note (TW): convert to visio drawing]

Figure 8-9 – 8x8 Scan Orders

Table 8-13 provides the mapping from the index idx of input list of 64 elements to indices i and j of the two-dimensional array c.

Table 8-13 – Specification of mapping of idx to c_{ij} for zig-zag and field scan

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
zig-zag	c_{00}	c_{01}	c_{10}	c_{20}	c_{11}	c_{02}	c_{03}	c_{12}	c_{21}	c_{30}	c_{40}	c_{31}	c_{22}	c_{13}	c_{04}	c_{05}
field	c_{00}	c_{10}	c_{20}	c_{01}	c_{11}	c_{30}	c_{40}	c_{21}	c_{02}	c_{31}	c_{50}	c_{60}	c_{70}	c_{41}	c_{12}	c_{03}

Table 8-13 – Specification of mapping of idx to cij for zig-zag and field scan (concluded)

idx	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
zig-zag	c_{14}	c_{23}	c_{32}	c_{41}	c_{50}	c_{60}	c_{51}	c_{42}	c_{33}	c_{24}	c_{15}	c_{06}	c_{07}	c_{16}	c_{25}	c_{34}
field	c_{22}	c_{51}	c_{61}	c_{71}	c_{32}	c_{13}	c_{04}	c_{23}	c_{42}	c_{52}	c_{62}	c_{72}	c_{33}	c_{14}	c_{05}	c_{24}

Table 8-13 – Specification of mapping of idx to cij for zig-zag and field scan (concluded)

idx	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
zig-zag	c_{43}	c_{52}	c_{61}	c_{70}	c_{71}	c_{62}	c_{53}	c_{44}	c_{35}	c_{26}	c_{17}	c_{27}	c_{36}	c_{45}	c_{54}	c_{63}
field	c_{43}	c_{53}	c_{63}	c_{73}	c_{34}	c_{15}	c_{06}	c_{25}	c_{44}	c_{54}	c_{64}	c_{74}	c_{35}	c_{16}	c_{26}	c_{45}

Table 8-13 – Specification of mapping of idx to cij for zig-zag and field scan (concluded)

idx	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
zig-zag	c_{72}	c_{73}	c_{64}	c_{55}	c_{46}	c_{37}	c_{47}	c_{56}	c_{65}	c_{74}	c_{75}	c_{66}	c_{57}	c_{67}	c_{76}	c_{77}
field	c_{55}	c_{65}	c_{75}	c_{36}	c_{07}	c_{17}	c_{46}	c_{56}	c_{66}	c_{76}	c_{27}	c_{37}	c_{47}	c_{57}	c_{67}	c_{77}

696938) Subclause 8.5.6 "Scaling and transformation process for luma DC transform coefficients for Intra_16x16 macroblock type"

Replace all occurrences of "QP_Y" in this subclause with "QP'_Y".

[Ed. Note: Fix the range of values allowed also.]

707039) Subclause 8.5.7 "Scaling and transformation process for chroma DC transform coefficients"

At the beginning of subclause 8.5.7 add the following

"The scaling and transformation process for chroma DC transform coefficients is specified as follows.

- If the chroma format is 4:2:0, the scaling and transformation process for chroma DC transform coefficients is specified in subclause 8.5.7.1.
- Otherwise, if the chroma format is 4:2:2, the scaling and transformation process for chroma DC transform coefficients is specified in subclause 8.5.7.2.
- Otherwise (the chroma format is 4:4:4), the scaling and transformation process for chroma DC transform coefficients is specified in subclause 8.5.7.3."

717140) New subclause 8.5.7.1 "Scaling and transformation process for chroma DC transform coefficients for 4:2:0 chroma format"

Place the entire current content of subclause 8.5.7 in a new subclause 8.5.7.1 "Scaling and transformation process for chroma DC transform coefficients for 4:2:0 chroma format".

Replace all occurrences of "QP_C" in this subclause with "QP'_C".

[Ed. Note: Fix the range of values allowed also: Each range stated as "-2^{15} to $2^{15}-1$" will need to be expanded to at least something like -2^(7+BitDepth) to 2^(7+BitDepth)-1, and consideration should be given to actual necessary range and range supported by CABAC and CAVLC.]

727241) New subclause 8.5.7.2 "Scaling and transformation process for chroma DC transform coefficients for 4:2:2 chroma format"

Inputs to this process are transform coefficient level values for chroma DC transform coefficients of one chroma component of the macroblock as a 2x4 array c with elements c_{ij}, where i and j form a two-dimensional frequency index.

Outputs of this process are 8 scaled DC values as a 2x4 array dcC with elements dcC_{ij}.

The inverse transform for the 2x4 chroma DC transform coefficients is specified by:

Error! Objects cannot be created from editing field codes.

[Ed. Note: Number the new equations.]

A bitstream conforming to this Recommendation | International Standard shall not contain data that results in any element fij of f that exceeds the range of integer values from -2^{15} to $2^{15}-1$, inclusive. [Ed. Note: That range and similar ranges will need to be expanded to at least something like -2^(7+BitDepth) to 2^(7+BitDepth)-1, and consideration should be given to actual necessary range and range supported by CABAC and CAVLC.]

The variable $QP'_{C,DC}$ is derived as

$$QP'_{C,DC} = QP'_{C} + 3$$

After the inverse transform, scaling is performed as follows.

– If $QP'_{C,DC}$ is greater than or equal to 12, the scaled result shall be derived as

$$dcC_{ij} = (f * LevelScale(QP'_{C,DC} \% 6, 0, 0, 0)) << (QP'_{C,DC} / 6 - 2), \text{ with } i = 0..3, j = 0, 1$$

– Otherwise ($QP'_{C,DC}$ is less than 12), the scaled result shall be derived as

$$dcC_{ij} = (f * LevelScale(QP'_{C,DC} \% 6, 0, 0, 0)) >> (2 - QP'_{C,DC} / 6), \text{ with } i = 0..3, j = 0, 1$$

A bitstream conforming to this Recommendation | International Standard shall not contain data that results in any element dcCij of dcC that exceeds the range of integer values from -2^{15} to $2^{15}-1$, inclusive. [Ed. Note: That range and similar ranges will need to be expanded to at least something like -2^(7+BitDepth) to 2^(7+BitDepth)-1, and consideration should be given to actual necessary range and range supported by CABAC and CAVLC.]

737342) New subclause 8.5.7.3 "Scaling and transformation process for chroma DC transform coefficients for 4:4:4 chroma format"

Inputs to this process are transform coefficient level values for chroma DC transform coefficients of one chroma component of the macroblock as a 4x4 array c with elements c_{ij}, where i and j form a two-dimensional frequency index.

Outputs of this process are 16 scaled DC values as a 4x4 array dcC with elements dcC_{ij}.

The inverse transform for the 4x4 chroma DC transform coefficients is specified by:

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

A bitstream conforming to this Recommendation | International Standard shall not contain data that results in any element fij of f that exceeds the range of integer values from -2^{15} to $2^{15}-1$, inclusive. [Ed. Note: That range and similar ranges will need to be expanded to at least something like -2^(7+BitDepth) to 2^(7+BitDepth)-1, and consideration should be given to actual necessary range and range supported by CABAC and CAVLC.]

After the inverse transform, scaling is performed as follows.

- If QP'_C is greater than or equal to 12, the scaled result shall be derived as

$$dcC_{ij} = (f * LevelScale(QP'_C \% 6, 0, 0, 0)) << (QP'_C / 6 - 2), \text{ with } i = 0..3, j = 0, 1$$

- Otherwise (QP'_C is less than 12), the scaled result shall be derived as

$$dcC_{ij} = (f * LevelScale(QP'_C \% 6, 0, 0, 0)) >> (2 - QP'_C / 6), \text{ with } i = 0..3, j = 0, 1$$

A bitstream conforming to this Recommendation | International Standard shall not contain data that results in any element dcC_{ij} of dcC that exceeds the range of integer values from -2^{15} to $2^{15}-1$, inclusive. [Ed. Note: That range and similar ranges will need to be expanded to at least something like -2^(7+BitDepth) to 2^(7+BitDepth)-1, and consideration should be given to actual necessary range and range supported by CABAC and CAVLC.]

747443) Subclause 8.5.8 "Scaling and transformation process for residual 4x4 blocks"

Replace "QP_Y" in this subclause with "QP'_Y".

Replace "QS_Y" in this subclause with "QS'_Y".

Replace "QP_C" in this subclause with "QP'_C".

Replace "QS_C" in this subclause with "QS'_C".

[Ed. Note: Define QS'_Y and QS'_C. in the same manner as QP'_Y and QP'_C.]

[Ed. Note: Fix the range of values allowed also.]

75) Subclause 8.5.10 Picture construction process prior to deblocking filter process

Change the title as follows.

"Picture construction process for 4x4 blocks prior to deblocking filter process"

76) New subclause 8.5.11 "Scaling and transformation process for residual 8x8 blocks"

Add a new subclause 8.5.11 as follows.

Input to this process is an 8x8 array c with elements c_{ij} which is an array relating to an 8x8 residual block of the luma component.

Outputs of this process are residual sample values as 8x8 array r with elements r_{ij}.

Scaling of 8x8 block transform coefficient levels c_{ij} proceeds as follows.

- If QP_Y is greater than or equal to 12, scaling of 8x8 block transform coefficient levels c_{ij} shall be performed as

$$d_{ij} = (c_{ij} * LevelScale64(QP_Y \%6, i, j)) << (QP_Y / 6 - 2), \quad \text{with} \quad i, j = 0..7 .$$

- Otherwise (QP_Y is less than 12), scaling of 8x8 block transform coefficient levels c_{ij} shall be performed as

$$d_{ij} = (c_{ij} * LevelScale64(QP_Y \%6, i, j) + 2^{1-QP_Y/6}) >> (2 - QP_Y / 6), \quad \text{with} \quad i, j = 0..7 .$$

The transform process shall convert the block of scaled transform coefficients to a block of output samples in a manner mathematically equivalent to the following.

First, each (horizontal) row of scaled transform coefficients is transformed using a one-dimensional inverse transform as follows.

A set of intermediate values e_{ij} is computed as follows.

$e_{i0} = d_{i0} + d_{i4}$, with i = 0..7 (Eq.-No.)

$e_{i1} = -d_{i3} + d_{i5} - d_{i7} - (d_{i7} >> 1)$, with i = 0..7 (Eq.-No.)

$e_{i2} = d_{i0} - d_{i4}$, with i = 0..7 (Eq.-No.)

$e_{i3} = d_{i1} + d_{i7} - d_{i3} - (d_{i3} >> 1)$, with i = 0..7 (Eq.-No.)

$e_{i4} = (d_{i2} >> 1) - d_{i6}$, with i = 0..7 (Eq.-No.)

$e_{i5} = -d_{i1} + d_{i7} + d_{i5} + (d_{i5} >> 1)$, with i = 0..7 (Eq.-No.)

$e_{i6} = d_{i2} + (d_{i6} >> 1)$, with i = 0..7 (Eq.-No.)

$e_{i7} = d_{i3} + d_{i5} + d_{i1} + (d_{i1} >> 1)$, with i = 0..7 (Eq.-No.)

A second set of intermediate results f_{ij} is computed from the intermediate values e_{ij} as follows.

$f_{i0} = e_{i0} + e_{i6}$, with i = 0..7 (Eq.-No.)

$f_{i1} = e_{i1} + (e_{i7} >> 2)$, with i = 0..7 (Eq.-No.)

$f_{i2} = e_{i2} + e_{i4}$, with i = 0..7 (Eq.-No.)

$f_{i3} = e_{i3} + (e_{i5} >> 2)$ with i = 0..7 (Eq.-No.)

$f_{i4} = e_{i2} - e_{i4}$, with i = 0..7 (Eq.-No.)

$f_{i5} = (e_{i3} >> 2) - e_{i5}$ with i = 0..7 (Eq.-No.)

$f_{i6} = e_{i0} - e_{i6}$, with i = 0..7 (Eq.-No.)

$f_{i7} = e_{i7} - (e_{i1} >> 2)$, with i = 0..7 (Eq.-No.)

Then, the transformed result g_{ij} is computed from these intermediate values f_{ij} as follows.

$g_{i0} = f_{i0} + f_{i7}$, with i = 0..7 (Eq.-No.)

$g_{i1} = f_{i2} + f_{i5}$, with i = 0..7 (Eq.-No.)

$g_{i2} = f_{i4} + f_{i3}$, with i = 0..7 (Eq.-No.)

$g_{i3} = f_{i6} + f_{i1}$, with i = 0..7 (Eq.-No.)

$g_{i4} = f_{i6} – f_{i1}$, with i = 0..7 (Eq.-No.)

$g_{i5} = f_{i4} – f_{i3}$, with i = 0..7 (Eq.-No.)

$g_{i6} = f_{i2} – f_{i5}$, with i = 0..7 (Eq.-No.)

$g_{i7} = f_{i0} – f_{i7}$, with i = 0..7 (Eq.-No.)

Then, each (vertical) column of the resulting matrix is transformed using the same one-dimensional inverse transform as follows.

A set of intermediate values h_{ij} is computed from the horizontally transformed value g_{ij} as follows.

$h_{i0} = g_{i0} + g_{i4}$, with i = 0..7 (Eq.-No.)

$h_{i1} = – g_{i3} + g_{i5} – g_{i7} – (g_{i7} >> 1)$, with i = 0..7 (Eq.-No.)

$h_{i2} = g_{i0} – g_{i4}$, with i = 0..7 (Eq.-No.)

$h_{i3} = g_{i1} + g_{i7} – g_{i3} – (g_{i3} >> 1)$, with i = 0..7 (Eq.-No.)

$h_{i4} = (g_{i2} >> 1) – g_{i6}$, with i = 0..7 (Eq.-No.)

$h_{i5} = – g_{i1} + g_{i7} + g_{i5} + (g_{i5} >> 1)$, with i = 0..7 (Eq.-No.)

$h_{i6} = g_{i2} + (g_{i6} >> 1)$, with i = 0..7 (Eq.-No.)

$h_{i7} = g_{i3} + g_{i5} + g_{i1} + (g_{i1} >> 1)$, with i = 0..7 (Eq.-No.)

A second set of intermediate results k_{ij} is computed from the intermediate values h_{ij} as follows.

$k_{i0} = h_{i0} + h_{i6}$, with i = 0..7 (Eq.-No.)

$k_{i1} = h_{i1} + (h_{i7} >> 2)$, with i = 0..7 (Eq.-No.)

$k_{i2} = h_{i2} + h_{i4}$, with i = 0..7 (Eq.-No.)

$k_{i3} = h_{i3} + (h_{i5} >> 2)$ with i = 0..7 (Eq.-No.)

$k_{i4} = h_{i2} – h_{i4}$, with i = 0..7 (Eq.-No.)

$k_{i5} = (h_{i3} >> 2) – h_{i5}$ with i = 0..7 (Eq.-No.)

$k_{i6} = h_{i0} – h_{i6}$, with i = 0..7 (Eq.-No.)

$k_{i7} = h_{i7} - (h_{i1} >> 2)$, with i = 0..7 (Eq.-No.)

Then, the transformed result m_{ij} is computed from these intermediate values k_{ij} as follows.

$m_{i0} = k_{i0} + k_{i7}$, with i = 0..7 (Eq.-No.)

$m_{i1} = k_{i2} + k_{i5}$, with i = 0..7 (Eq.-No.)

$m_{i2} = k_{i4} + k_{i3}$, with i = 0..7 (Eq.-No.)

$m_{i3} = k_{i6} + k_{i1}$, with i = 0..7 (Eq.-No.)

$m_{i4} = k_{i6} - k_{j1}$, with i = 0..7 (Eq.-No.)

$m_{i5} = k_{i4} - k_{i3}$, with i = 0..7 (Eq.-No.)

$m_{i6} = k_{i2} - k_{i5}$, with i = 0..7 (Eq.-No.)

$m_{i7} = k_{i0} - k_{i7}$, with i = 0..7 (Eq.-No.)

After performing both the one-dimensional horizontal and the one-dimensional vertical inverse transforms to produce an array of transformed samples, the final constructed residual sample values shall be derived as

$r_{ij} = (m_{ij} + 2^5) >> 6$ with i, j = 0..7 (Eq.-No.)

77) New subclause 8.5.12 "Picture construction process for 8x8 luma residuals prior to deblocking filter process"

Add a new subclause 8.5.12 as follows.

Inputs to this process are

- luma8x8BlkIdx
- a constructed residual sample 8x8 array u with elements u_{ij} which is a luma residual block
- the prediction sample 8x8 array $pred_L$

Outputs of this process are constructed luma sample blocks S'_L prior to the deblocking filter process.

The position of the upper-left luma sample of the current macroblock is derived by invoking the inverse macroblock scanning process in subclause 6.4.1 with CurrMbAddr as input and the output being assigned to (xP, yP).

For each sample u_{ij} of the 8x8 luma block, the following applies.

- The position of the upper-left sample of an 8x8 luma block with index luma8x8BlkIdx inside the macroblock is derived by invoking the inverse 8x8 luma block scanning process in subclause 6.4.2.2 with luma8x8BlkIdx as the input and the output being assigned to (xO, yO).
- Depending on the variable MbaffFrameFlag, the following applies.
  - If MbaffFrameFlag is equal to 1 and the current macroblock is a field macroblock

    $S'_L[xP + xO + j, yP + 2 * (yO + i)] = u_{ij}$ with i, j = 0..7 (Eq.-No)

  - Otherwise (MbaffFrameFlag is equal to 0 or the current macroblock is a frame macroblock),

$S'_L[\ xP + xO + j, yP + yO + i\ ] = u_{ij}$ with i, j = 0..7 (Eq.-No)

787844) Subclause 8.6.1.1 “Luma transform coefficient decoding process: Equation 8-296

In Equation 8-296, replace "Clip1" with "$Clip1_Y$".

797945) Subclause 8.6.1.2 “Chroma transform coefficient decoding process” Equation 8-303

In Equation 8-303, replace "Clip1" with "$Clip1_C$".

808046) Subclause 8.6.2.1 “Luma transform coefficient decoding process” Equation 8-312

In Equation 8-312, replace "Clip1" with "$Clip1_Y$".

818147) Subclause 8.6.2.2 “Chroma transform coefficient decoding process” Equation 8-316

In Equation 8-316, replace "Clip1" with "$Clip1_C$".

828248) Prediction and Transform modifications for 4:2:2 and 4:4:4 formats [Ed. Note: Needs format work]

838349) Inter Prediction

4:4:4 Same motion vectors are used directly for all components. Use 6 tap filters for all components.

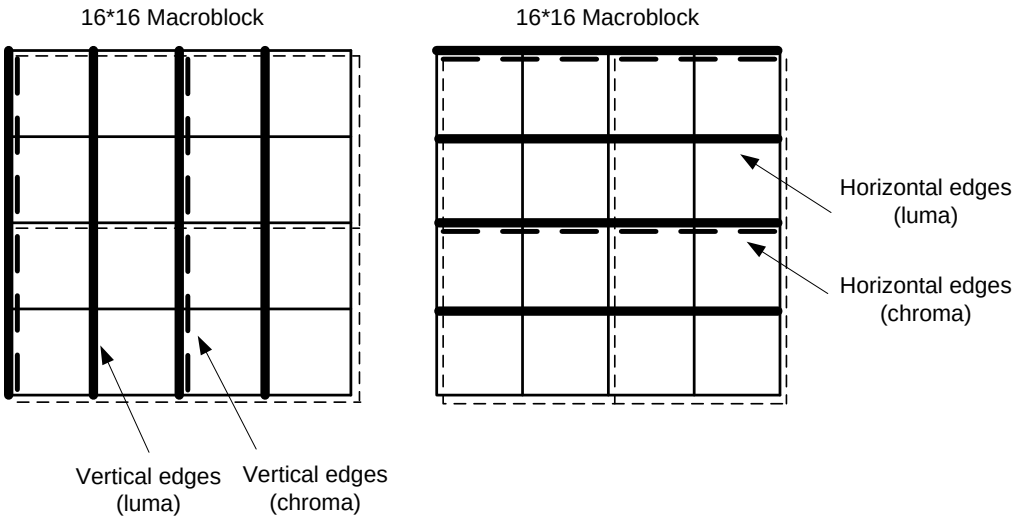
4:2:2 Vertical motion vectors are used directly. Horizontal vectors are scaled by 2 as for 4:2:0. Use bilinear interpolation based on 1/8 sample horizontal and 1/4 sample vertical motion resolution.

848450) Deblocking modifications

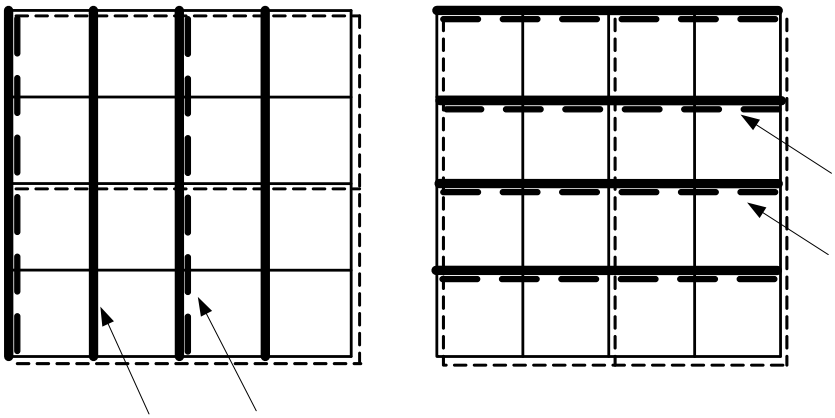
[Ed: Refine below.]

Deblocking filter process

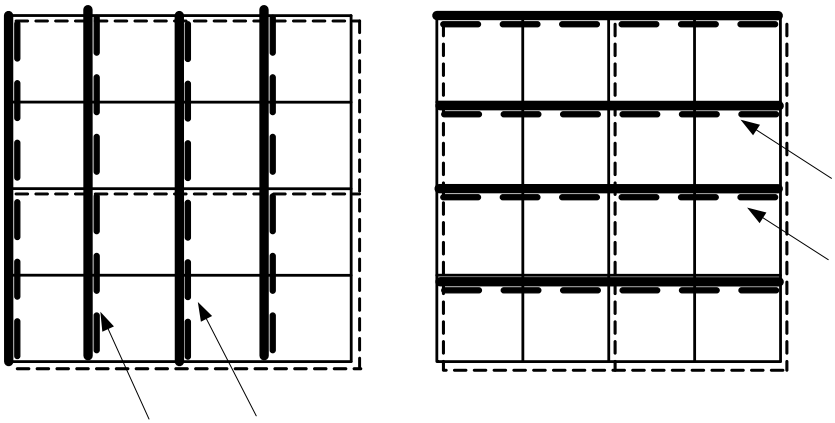
The deblocking filter process is invoked for the luma and chroma components separately. For each macroblock, vertical edges are filtered first, from left to right, and then horizontal edges are filtered from top to bottom. The luma deblocking filter process is performed on four 16-sample edges, and the deblocking filter process for each chroma components is performed on two 8-sample edges when chroma_format is 1 (4:2:0 format), and on four 16-sample edges when chroma_format is 3 (4:4:4 format), for the horizontal direction as shown on the left side of Figure 8-9 and for the vertical direction as shown on the right side of Figure 8-9.



(a) 4:2:0 format



(b) 4:2:2 format



(c) 4:4:4 format

Figure 8-9 – Boundaries in a macroblock to be filtered (luma boundaries shown with solid lines and chroma boundaries shown with dashed lines)

Subclause: 8.7.1 Filtering process for block edges

…

The variable nE is derived as follows.

If chromaEdgeFlag is equal to 0, nE is 16;

If chromaEdgeFlag is equal to 1 and chroma_format is 3, nE is 16;

Otherwise (chromaEdgeFlag is equal to 1 and chroma_format is 1), nE is 8.

Subclauses: 8.7.2.2 and 8.7.2.3, Loop filter parameters must be scaled to BitDepth

Multiply the values of α and β in Table 8-14 and the value of t_{C0} from Table 8-15 by (1 << (BitDepth – 8)). [Ed. Better to define a separate variable such as defining Table 8-14 to contain α' and β', and define α = α' * (1 << (BitDepth – 8)) and β' = β' * (1 << (BitDepth – 8)).]

85) Subclause 8.7 "Deblocking filter process" [Ed. Note: Conflicting Edits]

Change subclause 8.7 as follows.

Change the first sentence of the first paragraph as follows.

A conditional filtering shall be applied to all ~~4x4~~ NxN (where N = 4 or N = 8 for luma, and N = 4 for chroma) block edges of a picture, except edges at the boundary of the picture and any edges for which the deblocking filter process is disabled by disable_deblocking_filter_idc, as specified below.

Change the second sentence of the second paragraph as follows.

The deblocking filter process is invoked for the luma and chroma components separately. For each macroblock, vertical edges are filtered first, from left to right, and then horizontal edges are filtered from top to bottom. The luma deblocking filter process is performed on two (when N = 8) or four (when N = 4) 16-sample edges and the deblocking filter process for each chroma components is performed on two 8-sample edges, for the horizontal direction as shown on the left side of Figure 8-9 and for the vertical direction as shown on the right side of Figure 8-9.#

Change Figure 8-9 as follows.

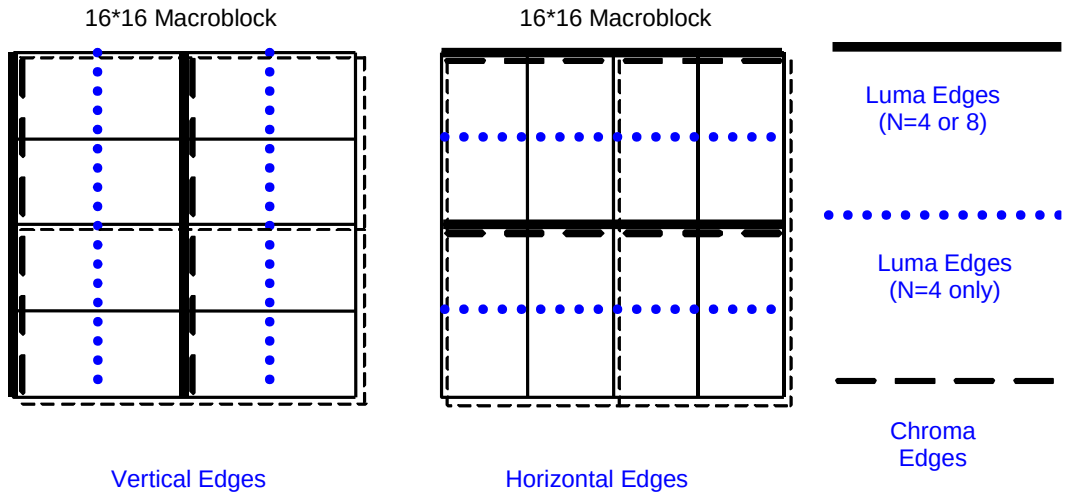


Figure 8-9 – Boundaries in a macroblock to be filtered (luma boundaries shown with solid lines for N=4 and N=8, luma boundaries shown with dotted lines for N=4 and chroma boundaries shown with dashed lines)

Change the text below Figure 8-9 as follows.

For each macroblock in ascending order of mbAddr, the following applies.

1. The variables fieldModeMbFlag, filterNon8x8LumaEdgesFlag, filterInternalEdgesFlag, filterLeftMbEdgeFlag and filterTopMbEdgeFlag are derived as follows.
   - The variable fieldModeMbFlag is derived as follows.
     - If any of the following conditions is true, fieldModeMbFlag is set equal to 1.
       - field_pic_flag is equal to 1
       - MbaffFrameFlag is equal 1 and the macroblock mbAddr is a field macroblock
     - Otherwise, fieldModeMbFlag is set equal to 0.
   - The variable filterNon8x8LumaEdgesFlag is derived as follows.
     - If any of the following conditions is true, filterNon8x8LumaEdgesFlag is set equal to 1.
       - transform_8x8_mode_flag is equal to 0
       - NoMbPartLessThan8x8Flag is equal to 0
       - transform_size_flag is equal to 0
       - mb_type for the macroblock mbAddr is equal to Intra_16x16
     - Otherwise, filterNon8x8LumaEdgesFlag is set equal to 0.
   - The variable filterInternalEdgesFlag is derived as follows.
     - If disable_deblocking_filter_idc for the slice that contains the macroblock mbAddr is equal to 1, the variable filterInternalEdgesFlag is set equal to 0;
     - Otherwise (disable_deblocking_filter_idc for the slice that contains the macroblock mbAddr is not equal to 1), the variable filterInternalEdgesFlag is set equal to 1.
   - The variable filterLeftMbEdgeFlag is derived as follows.
     - If any of the following conditions is true, the variable filterLeftMbEdgeFlag is set equal to 0.
       - the left vertical macroblock edge of the macroblock mbAddr represents a picture boundary

- disable_deblocking_filter_idc for the slice that contains the macroblock mbAddr is equal to 1
- disable_deblocking_filter_idc for the slice that contains the macroblock mbAddr is equal to 2 and the left vertical macroblock edge of the macroblock mbAddr represents a slice boundary

- Otherwise, the variable filterLeftMbEdgeFlag is set equal to 1.

- The variable filterTopMbEdgeFlag is derived as follows.
  - If any of the following conditions is true, the variable filterTopMbEdgeFlag is set equal to 0.
    - the top horizontal macroblock edge of the macroblock mbAddr represents a picture boundary
    - disable_deblocking_filter_idc for the slice that contains the macroblock mbAddr is equal to 1
    - disable_deblocking_filter_idc for the slice that contains the macroblock mbAddr is equal to 2 and the top horizontal macroblock edge of the macroblock mbAddr represents a slice boundary
  - Otherwise, the variable filterTopMbEdgeFlag is set equal to 1.

2. Given the variables fieldModeMbFlag, filterNon8x8LumaEdgesFlag, filterInternalEdgesFlag, filterLeftMbEdgeFlag and filterTopMbEdgeFlag the deblocking filtering is controlled as follows.
   - When filterLeftMbEdgeFlag is equal to 1, the filtering of the left vertical luma edge is specified as follows.
     - The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (0, k)$ with k = 0..15 as input and S'_L as output.
   - When filterInternalEdgesFlag is equal to 1, the filtering of the internal vertical luma edges is specified as follows.
     - If filterNon8x8LumaEdgesFlag is equal to 1, The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (4, k)$ with k = 0..15 as input and S'_L as output.
     - The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (8, k)$ with k = 0..15 as input and S'_L as output.
     - If filterNon8x8LumaEdgesFlag is equal to 1, The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (12, k)$ with k = 0..15 as input and S'_L as output.
   - When filterTopMbEdgeFlag is equal to 1, the filtering of the top horizontal luma edge is specified as follows.
     - If MbaffFrameFlag is equal to 1, (mbAddr % 2) is equal to 0, mbAddr is greater than or equal to 2 * PicWidthInMbs, the macroblock mbAddr is a frame macroblock, and the macroblock (mbAddr - 2 * PicWidthInMbs + 1) is a field macroblock, the following applies.
       - The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = 1, and $(xE_k, yE_k) = (k, 0)$ with k = 0..15 as input and S'_L as output.
       - The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = 1, and $(xE_k, yE_k) = (k, 1)$ with k = 0..15 as input and S'_L as output.
     - Otherwise, the process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (k, 0)$ with k = 0..15 as input and S'_L as output.
   - When filterInternalEdgesFlag is equal to 1, the filtering of the internal horizontal luma edges is specified as follows.
     - If filterNon8x8LumaEdgesFlag is equal to 1, The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (k, 4)$ with k = 0..15 as input and S'_L as output.
     - The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (k, 8)$ with k = 0..15 as input and S'_L as output.

- If filterNon8x8LumaEdgesFlag is equal to 1, The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (k, 12)$ with k = 0..15 as input and S'_L as output.

86) Subclause 9.3.1.1 "Initialisation process for context variables"

Change subclause 9.3.1.1 as follows.

Change Table 9-11 with the following.

	Syntax element	Table	Slice type			
			SI	I	P, SP	B
slice_data( )	mb_skip_flag	Table 9-13 Table 9-14			11-13	24-26
	mb_field_decoding_flag	Table 9-18	70-72	70-72	70-72	70-72
macroblock_layer( )	mb_type	Table 9-12, Table 9-13, Table 9-14.	0-10	3-10	14-20	27-35
	transform_size_flag	Table 9-16	na	399-401	399-401	399-401
	coded_block_pattern (luma)	Table 9-18	73-76	73-76	73-76	73-76
	coded_block_pattern (chroma)	Table 9-18	77-84	77-84	77-84	77-84
	mb_qp_delta	Table 9-17	60-63	60-63	60-63	60-63
mb_pred( )	prev_intra4x4_pred_mode_flag	Table 9-17	68	68	68	68
	rem_intra4x4_pred_mode	Table 9-17	69	69	69	69
	prev_intra8x8_pred_mode_flag	Table 9-17	na	68	68	68
	rem_intra8x8_pred_mode	Table 9-17	na	69	69	69
	intra_chroma_pred_mode	Table 9-17	64-67	64-67	64-67	64-67
mb_pred( ) and sub_mb_pred( )	ref_idx_l0	Table 9-16			54-59	54-59
	ref_idx_l1	Table 9-16				54-59
	mvd_l0[ ][ ][0]	Table 9-15			40-46	40-46
	mvd_l1[ ][ ][0]	Table 9-15				40-46
	mvd_l0[ ][ ][1]	Table 9-15			47-53	47-53
	mvd_l1[ ][ ][1]	Table 9-15				47-53
sub_mb_pred( )	sub_mb_type	Table 9-13 Table 9-14			21-23	36-39
residual_block_cabac( )	coded_block_flag	Table 9-18	85-104	85-104	85-104	85-104

	significant_coeff_flag[ ]	Table 9-19, Table 9-22, Table 9-XX, Table 9-XX	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450
	last_significant_coeff_flag[ ]	Table 9-20, Table 9-23, Table 9-XX, Table 9-XX.	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459
	coeff_abs_level_minus1[ ]	Table 9-21, Table 9-XX	227-275 426-435	227-275 426-435	227-275 426-435	227-275 426-435

Change Table 9-16 as follows.

Table 9-16 – Values of variables m and n for ctxIdx from 54 to 59, and 399 to 401 [Ed. (DM): initialisation values for two cabac_init_idc cases may be changed]

Value of cabac_init_idc	Initialisation variables	ctxIdx								
		54	55	56	57	58	59	399	400	401
I slices	m	na	na	na	na	na	na	0	0	0
	n	na	na	na	na	na	na	41	63	63
0	m	-7	-5	-4	-5	-7	1	0	0	0
	n	67	74	74	80	72	58	41	63	63
1	m	-1	-1	1	-2	-5	0	0	0	0
	n	66	77	70	86	72	61	41	63	63
2	m	3	-4	-2	-12	-7	1	0	0	0
	n	55	79	75	97	50	60	41	63	63

Insert a Table as follows.

Table 9-XX – Values of variables m and n for ctxIdx from 402 to 459 [Ed. (DM): initialisation values for two cabac_init_idc cases may be further refined]

ctxIdx	I slices		Value of cabac_init_idc						ctxIdx	I slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
402	-1	73	-4	60	-4	60	-4	60	431	-4	56	-7	54	-7	54	-7	54
403	-7	73	-3	49	-3	49	-3	49	432	-1	59	-2	58	-2	58	-2	58
404	-6	76	-2	50	-2	50	-2	50	433	-6	71	-4	63	-4	63	-4	63
405	-7	71	-4	49	-4	49	-4	49	434	-8	74	-5	66	-5	66	-5	66
406	-9	72	-5	48	-5	48	-5	48	435	-11	85	1	64	1	64	1	64
407	-5	65	-2	46	-2	46	-2	46	436	-1	73	-4	60	-4	60	-4	60
408	-14	83	-7	54	-7	54	-7	54	437	-7	73	-3	49	-3	49	-3	49
409	-8	72	-1	45	-1	45	-1	45	438	-6	76	-2	50	-2	50	-2	50
410	-10	75	-4	49	-4	49	-4	49	439	-7	71	-4	49	-4	49	-4	49
411	-5	64	4	39	4	39	4	39	440	-9	72	-5	48	-5	48	-5	48
412	-4	59	0	42	0	42	0	42	441	-5	65	-2	46	-2	46	-2	46
413	-13	79	2	43	2	43	2	43	442	-14	83	-7	54	-7	54	-7	54
414	-9	69	0	44	0	44	0	44	443	-8	72	-1	45	-1	45	-1	45
415	-8	66	5	32	5	32	5	32	444	-10	75	-4	49	-4	49	-4	49
416	3	55	15	30	15	30	15	30	445	-5	64	4	39	4	39	4	39
417	12	33	17	27	17	27	17	27	446	-4	59	0	42	0	42	0	42
418	5	38	23	13	23	13	23	13	447	-13	79	2	43	2	43	2	43
419	9	34	24	16	24	16	24	16	448	-9	69	0	44	0	44	0	44
420	18	22	22	25	22	25	22	25	449	-8	66	5	32	5	32	5	32
421	19	22	23	27	23	27	23	27	450	3	55	15	30	15	30	15	30
422	23	19	23	32	23	32	23	32	451	12	33	17	27	17	27	17	27
423	26	16	17	43	17	43	17	43	452	5	38	23	13	23	13	23	13
424	14	44	17	49	17	49	17	49	453	9	34	24	16	24	16	24	16
425	40	14	2	70	2	70	2	70	454	18	22	22	25	22	25	22	25
426	-9	75	-3	58	-3	58	-3	58	455	19	22	23	27	23	27	23	27
427	-1	44	-1	28	-1	28	-1	28	456	23	19	23	32	23	32	23	32
428	-2	49	0	29	0	29	0	29	457	26	16	17	43	17	43	17	43
429	-2	51	2	30	2	30	2	30	458	14	44	17	49	17	49	17	49
430	-1	51	1	35	1	35	1	35	459	40	14	2	70	2	70	2	70

87) Subclause 9.3.2 "Binarisation Process"

Change the following praragraph in subclause 9.3.2 as follows.

The possible values of the context index ctxIdx are in the range 0 to ~~398~~459, inclusive. The value assigned to ctxIdxOffset specifies the lower value of the range of ctxIdx assigned to the corresponding binarisation or binarisation part of a syntax element.

88~~88~~51) Table 9-4 – Assignment of codeNum to values of coded_block_pattern for macroblock prediction modes

Replace Table 9-4 as follows.

Table 9-4 – Assignment of codeNum to values of coded_block_pattern for macroblock prediction modes

(a) chroma_format_idc!=0

codeNum	coded_block_pattern	
	Intra_4x4	Inter
0	47	0
1	31	16
2	15	1
3	0	2
4	23	4
5	27	8
6	29	32
7	30	3
8	7	5
9	11	10
10	13	12
11	14	15
12	39	47
13	43	7
14	45	11
15	46	13
16	16	14
17	3	6
18	5	9
19	10	31

20	12	35
21	19	37
22	21	42
23	26	44
24	28	33
25	35	34
26	37	36
27	42	40
28	44	39
29	1	43
30	2	45
31	4	46
32	8	17
33	17	18
34	18	20
35	20	24
36	24	19
37	6	21
38	9	26
39	22	28
40	25	23
41	32	27
42	33	29
43	34	30
44	36	22
45	40	25
46	38	38
47	41	41

(b) chroma_format_idc=0

codeNum	**coded_block_pattern**	
	Intra_4x4	**Inter**
0	15	0
1	0	16
2	7	1

3	11	2
4	13	4
5	14	8
6	16	3
7	3	5
8	5	10
9	10	12
10	12	15
11	1	7
12	2	11
13	4	13
14	8	14
15	6	6
16	9	9

898952) Table 9-6 - Codeword table for level_prefix

Substitute the following for Table 9-6 - Codeword table for level_prefix:

level_prefix	bit string
0	1
1	01
2	001
3	0001
4	0000 1
5	0000 01
6	0000 001
7	0000 0001
8	0000 0000 1
9	0000 0000 01
10	0000 0000 001
11	0000 0000 0001
12	0000 0000 0000 1
13	0000 0000 0000 01
14	0000 0000 0000 001
15	0000 0000 0000 0001
16	0000 0000 0000 0000 1
17	0000 0000 0000 0000 01
18	0000 0000 0000 0000 001
19	0000 0000 0000 0000 0001
20	0000 0000 0000 0000 0000 1
…	…

909053) Subclause 9.2.2 "Parsing process for level information"

Substitute the following for 9.2.2:

Inputs to this process are bits from slice data, the number of non-zero transform coefficient levels TotalCoeff(coeff_token), and the number of trailing one transform coefficient levels TrailingOnes(coeff_token).

Output of this process is a list with name level containing transform coefficient levels.

Initially an index i is set equal to 0. Then the following procedure is iteratively applied TrailingOnes(coeff_token) times to decode the trailing one transform coefficient levels (if any):

- A 1-bit syntax element trailing_ones_sign_flag is decoded and evaluated as follows.
  - If trailing_ones_sign_flag is equal to 0, the value +1 is assigned to level[i].
  - Otherwise (trailing_ones_sign_flag is equal to 1), the value -1 is assigned to level[i].
- The index i is incremented by 1.

Following the decoding of the trailing one transform coefficient levels, a variable suffixLength is initialised as follows.

- If TotalCoeff(coeff_token) is greater than 10 and TrailingOnes(coeff_token) is less than 3, suffixLength is set equal to 1.
- Otherwise (TotalCoeff(coeff_token) is less than or equal to 10 or TrailingOnes(coeff_token) is equal to 3), suffixLength is set equal to 0.

The following procedure is then applied iteratively (TotalCoeff(coeff_token) – TrailingOnes(coeff_token)) times to decode the remaining levels (if any):

- The syntax element level_prefix is decoded using the VLC specified in Table 9-6.
- The variable levelSuffixSize is set equal to the variable suffixLength with the exception of the following two cases.
- When level_prefix is equal to 14 and suffixLength is equal to 0, levelSuffixSize is set equal to 4.
- When level_prefix is greater than or equal to 15, levelSuffixSize is set equal to level_prefix - 3.
- The syntax element level_suffix is decoded as follows.
  - If levelSuffixSize is greater than 0, the syntax element level_suffix is decoded as unsigned integer representation u(v) with levelSuffixSize bits.
  - Otherwise (levelSuffixSize is equal to 0), the syntax element level_suffix shall be inferred to be equal to 0.
- A variable levelCode is set equal to (min(15,level_prefix) << suffixLength) + level_suffix.
- When level_prefix is greater than or equal to 15 and suffixLength is equal to 0, levelCode is incremented by 15.
- When level_prefix is greater than or equal to 16, levelCode is incremented by (1<<(level_prefix – 3)) – 4096.
- When the index i is equal to TrailingOnes(coeff_token) and TrailingOnes(coeff_token) is smaller than 3, levelCode is incremented by 2.
- The variable level[i] is derived as follows.
  - If levelCode is an even number, the value (levelCode + 2) >> 1 is assigned to level[i].
  - Otherwise, the value (-levelCode – 1) >> 1 is assigned to level[i].
- When suffixLength is equal to 0, suffixLength is set equal to 1.
- When the absolute value of level[i] is greater than (3 << (suffixLength – 1)) and suffixLength is less than 6, suffixLength is incremented by 1.
- The index i is incremented by 1.

91) CABAC Support for 4:2:2 or 4:4:4

Since the chroma DC blocks will change in dimension from 2x2 to 2x4 or 4x4, coding of the chroma DC transform coefficients has to be adapted. The flags significant_coeff_flag and last_siginificant_coeff_flag are encoded using a model related to their position in the scanning path. If we want to keep that modeling scheme, 2x4 or 2x8 additional models would be required for each flag. Furthermore, for the coding of the absolute values of chroma DC levels one additional model would be necessary, since the number of decoded levels with abs. value greater than 1 can be larger than 3 in chroma DC blocks with dimension 2x4 or 4x4.

929254) Subclause 9.3 "CABAC parsing process for slice data"

*In subclause 9.3, replace the phrase "*the pcm_alignment_zero_bit and all pcm_byte data*" with "*any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data*".*

939355) Subclause 9.3.1 "Initialiszation process"

*In subclause 9.3.1, replace the phrase "*the pcm_alignment_zero_bit and all pcm_byte data*" with "*any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data*".*

949456) Subclause 9.3.1.2 "Initialiszation process for the arithmetic decoding engine"

*In subclause 9.3.1.2, replace the phrase "*the pcm_alignment_zero_bit and all pcm_byte data*" with "*any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data*".*

959557) Subclause 9.3.4.1 "Initialisation process for the arithmetic encoding engine (informative)"

*In subclause 9.3.4.1, replace the phrase "*the pcm_alignment_zero_bit and all pcm_byte data*" with "*any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data*".*

969658) Entropy coding modifications for chroma extensions.

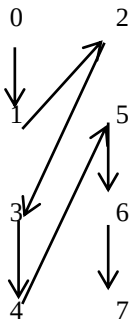
The only changes needed are for coding of chroma DC coefficients. In addition the number of AC blocks will be different, but coding of each AC 4x4 block is unchanged.

979759) VLC Modification for 4:2:2

The table total_ceff()/trailing_ones():Num-VLC_Chroma_DC must be changed reflecting that there may be 0 to 8 coefficients. New code table:

Trailing_ones	0	1	2	3
Total_coeff				
0	1	-	-	-
1	0001111	01	-	-
2	0001110	0001101	001	-
3	000000111	0001100	000101	00001
4	000000110	000000101	0001010	000001
5	0000000111	0000000110	000000100	0001001
6	00000000111	00000000110	0000000101	0001000
7	000000000111	000000000110	00000000101	0000000100
8	0000000000111	000000000101	000000000100	00000000100

A scanning order for the 2x4 transform must be defined and the following is used:



The table for total_zeroes_chroma_DC must be changed reflecting that there may be 0 to 7 zeroes. New code table:

NumCoeff	1	2	3	4	5	6	7	
Total zeroes								
0	1	000	000		110	00	00	0
1	010	01	001		00	01	01	1
2	011	001	01		01	10	1	
3	0010		100		10	10	11	
4	0011		101		110	111		
5	0001		110		111			
6	00001		111					
7	00000							

98~~60~~) VLC Modification for 4:4:4

Similar action is taken here. However, suitable tables already exist for 4x4 blocks.

Use total_ceff()/trailing_ones():Num-VLC0

Use normal Zig-zag scanning

Use the same table for total_zeroes as for luma

99) [Ed note on new constraints for existing profiles.]

For Main, Baseline, and Extended profiles, specify constrants as follows.

– *No 8x8 transform*

– *No new lossless coding feature*

– *No quantisation matrices*

– Sequence parameter sets shall have chroma_format_idc equal to 1 (inferred).

– Sequence parameter sets shall have bit_depth_luma_minus8 equal to 0 (inferred).

– Sequence parameter sets shall have bit_depth_chroma_minus8 equal to 0 (inferred).

~~61) CABAC Support for 4:2:2 or 4:4:4~~

~~Since the chroma DC blocks will change in dimension from 2x2 to 2x4 or 4x4, coding of the chroma DC transform coefficients has to be adapted. The flags significant_coeff_flag and last_siginificant_coeff_flag are encoded using a model related to their position in the scanning path. If we want to keep that modeling scheme, 2x4 or 2x8 additional models would be required for each flag. Furthermore, for the coding of the absolute values of chroma DC levels one additional model would be necessary, since the number of decoded levels with abs. value greater than 1 can be larger than 3 in chroma DC blocks with dimension 2x4 or 4x4.~~

100~~10062~~) New subclause A.2.4 "4:2:0/10 profile"

Insert new subclause A.2.4 as follows

A.2.4 4:2:0/10 profile

Bitstreams conforming to the 4:2:2/10 profile shall obey the following constraints:

- Only I, P, and B slice types may be present.
- NAL unit streams shall not contain nal_unit_type values in the range of 2 to 4, inclusive.
- Arbitrary slice order is not allowed.
- Picture parameter sets shall have num_slice_groups_minus1 equal to 0 only.
- Picture parameter sets shall have redundant_pic_cnt_present_flag equal to 0 only.
- Sequence parameter sets shall have chroma_format_idc in the range of 0 to 1 inclusive.
- Sequence parameter sets shall have bit_depth_luma_minus8 in the range of 0 to 2 inclusive.
- Sequence parameter sets shall have bit_depth_chroma_minus8 in the range of 0 to 2 inclusive.
- Sequence parameter sets shall have lossless_qp0~~coding~~_flag equal to 0 only.
- The level constraints specified for the 4:2:2/10 profile in subclause A.3 [Ed. ???] shall be fulfilled.

Conformance of a bitstream to the 4:2:0/10 profile is specified by profile_idc being equal to 69. Decoders conforming to the 4:2:2/10 profile at a specific level shall be capable of decoding all bitstreams in which profile_idc is equal to 69 or 66 or constraint_set1_flag is equal to 1, or both constraint_set3_flag is equal to 1 and level_idc is not equal to 11, and in which level_idc represents a level less than or equal to the specified level. [Ed. Note: Use parenthesis above to group the both?]

101~~10163~~) New subclause A.2.5 "4:2:2/8 profile"

Insert new subclause A.2.5 as follows

A.2.5 4:2:2/8 profile

Bitstreams conforming to the 4:2:2/8 profile shall obey the following constraints:

- Only I, P, and B slice types may be present.
- NAL unit streams shall not contain nal_unit_type values in the range of 2 to 4, inclusive.
- Arbitrary slice order is not allowed.
- Picture parameter sets shall have num_slice_groups_minus1 equal to 0 only.
- Picture parameter sets shall have redundant_pic_cnt_present_flag equal to 0 only.
- Sequence parameter sets shall have chroma_format_idc in the range of 0 to 2 inclusive.
- Sequence parameter sets shall have bit_depth_luma_minus8 equal to 0 only.

– Sequence parameter sets shall have bit_depth_chroma_minus8 equal to 0 only.
– Sequence parameter sets shall have lossless_qp0~~coding~~_flag equal to 0 only.
– The level constraints specified for the 4:2:2/8 profile in subclause A.3 [Ed. ???] shall be fulfilled.

Conformance of a bitstream to the 4:2:2/8 profile is specified by profile_idc being equal to 70. Decoders conforming to the 4:2:2/8 profile at a specific level shall be capable of decoding all bitstreams in which profile_idc is equal to 70 or 66 or constraint_set1_flag is equal to 1, or both constraint_set3_flag is equal to 1 and level_idc is not equal to 11, and in which level_idc represents a level less than or equal to the specified level.

102~~10264~~) New subclause A.2.4 "4:2:2/10 profile"

Insert new subclause A.2.4 as follows

A.2.44:2:2/10 profile

Bitstreams conforming to the 4:2:2/10 profile shall obey the following constraints:

– Only I, P, and B slice types may be present.
– NAL unit streams shall not contain nal_unit_type values in the range of 2 to 4, inclusive.
– Arbitrary slice order is not allowed.
– Picture parameter sets shall have num_slice_groups_minus1 equal to 0 only.
– Picture parameter sets shall have redundant_pic_cnt_present_flag equal to 0 only.
– Sequence parameter sets shall have chroma_format_idc in the range of 0 to 2 inclusive
– Sequence parameter sets shall have bit_depth_luma_minus8 in the range of 0 to 2 inclusive.
– Sequence parameter sets shall have bit_depth_chroma_minus8 in the range of 0 to 2 inclusive.
– Sequence parameter sets shall have lossless_qp0~~coding~~_flag equal to 0 only.
– The level constraints specified for the 4:2:2/10 profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the 4:2:2/10 profile is specified by profile_idc being equal to 71. Decoders conforming to the 4:2:2/10 profile at a specific level shall be capable of decoding all bitstreams in which profile_idc is equal to 71 , 70, 69 or 66 or constraint_set1_flag is equal to 1 and in which level_idc represents a level less than or equal to the specified level.

103~~10265~~) New subclause A.2.5 "4:4:4/12 profile"

Insert new subclause A.2.5 as follows

A.2.5 4:4:4/12 profile

Bitstreams conforming to the 4:4:4/12 profile shall obey the following constraints:

– Only I, P, and B slice types may be present.
– NAL unit streams shall not contain nal_unit_type values in the range of 2 to 4, inclusive.
– Arbitrary slice order is not allowed.
– Picture parameter sets shall have num_slice_groups_minus1 equal to 0 only.
– Picture parameter sets shall have redundant_pic_cnt_present_flag equal to 0 only.
– Sequence parameter sets shall have chroma_format_idc in the range of 0 to 3 inclusive
– Sequence parameter sets shall have bit_depth_luma_minus8 in the range of 0 to 4 inclusive.
– Sequence parameter sets shall have bit_depth_chroma_minus8 in the range of 0 to 4 inclusive.
– The level constraints specified for the 4:4:4/12 profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the 4:4:4/12 profile is specified by profile_idc being equal to 72. Decoders conforming to the 4:4:4/12 profile at a specific level shall be capable of decoding all bitstreams in which profile_idc is equal to 66, 69,

70, 71, or 72 or constraint_set1_flag is equal to 1 and in which level_idc represents a level less than or equal to the specified level.

62) New subclause A.2.4 "4:2:2/10 profile"

Insert new subclause A.2.4 as follows

A.2.4 4:2:2/10 profile

Bitstreams conforming to the 4:2:2/10 profile shall obey the following constraints:

- Only I, P, and B slice types may be present.
- NAL unit streams shall not contain nal_unit_type values in the range of 2 to 4, inclusive.
- Arbitrary slice order is not allowed.
- Picture parameter sets shall have num_slice_groups_minus1 equal to 0 only.
- Picture parameter sets shall have redundant_pic_cnt_present_flag equal to 0 only.
- Sequence parameter sets shall have chroma_format_idc in the range of 0 to 2 inclusive
- Sequence parameter sets shall have bit_depth_luma_minus8 in the range of 0 to 2 inclusive.
- Sequence parameter sets shall have bit_depth_chroma_minus8 in the range of 0 to 2 inclusive.
- The level constraints specified for the 4:2:2/10 profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the 4:2:2/10 profile is specified by profile_idc being equal to 69. Decoders conforming to the 4:2:2/10 profile at a specific level shall be capable of decoding all bitstreams in which profile_idc is equal to 69 or 66 or constraint_set1_flag or constraint_set4_flag is equal to 1 and in which level_idc represents a level less than or equal to the specified level.

[Ed. Note: What about constraint_setN_flag? Creating a new one for each profile?]

63) New subclause A.2.5 "4:4:4/12 profile"

Insert new subclause A.2.5 as follows

A.2.5 4:4:4/12 profile

Bitstreams conforming to the 4:4:4/12 profile shall obey the following constraints:

- Only I, P, and B slice types may be present.
- NAL unit streams shall not contain nal_unit_type values in the range of 2 to 4, inclusive.
- Arbitrary slice order is not allowed.
- Picture parameter sets shall have num_slice_groups_minus1 equal to 0 only.
- Picture parameter sets shall have redundant_pic_cnt_present_flag equal to 0 only.
- Sequence parameter sets shall have chroma_format_idc in the range of 0 to 3 inclusive
- Sequence parameter sets shall have bit_depth_luma_minus8 in the range of 0 to 4 inclusive.
- Sequence parameter sets shall have bit_depth_chroma_minus8 in the range of 0 to 4 inclusive.
- The level constraints specified for the 4:4:4/12 profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the 4:4:4/12 profile is specified by profile_idc being equal to 70. Decoders conforming to the 4:4:4/12 profile at a specific level shall be capable of decoding all bitstreams in which profile_idc is equal to 66, 69, or 70 or constraint_set1_flag or constraint_set4_flag is equal to 1 and in which level_idc represents a level less than or equal to the specified level.

[Ed. Note: What about constraint_setN_flag? Creating a new one for each profile?]

~~**64) New subclause A.2.6 "4:2:2/8 profile"**~~

~~*Insert new subclause A.2.6 as follows*~~

~~**A.2.6 4:4:4/8 profile**~~

~~Bitstreams conforming to the 4:4:4/8 profile shall obey the following constraints:~~

- ~~Only I, P, and B slice types may be present.~~
- ~~NAL unit streams shall not contain nal_unit_type values in the range of 2 to 4, inclusive.~~
- ~~Arbitrary slice order is not allowed.~~
- ~~Picture parameter sets shall have num_slice_groups_minus1 equal to 0 only.~~
- ~~Picture parameter sets shall have redundant_pic_cnt_present_flag equal to 0 only.~~
- ~~Sequence parameter sets shall have chroma_format_idc in the range of 0 to 3 inclusive~~
- ~~Sequence parameter sets shall have bit_depth_luma_minus8 equal to 0.~~
- ~~Sequence parameter sets shall have bit_depth_chroma_minus8 equal to 0.~~
- ~~The level constraints specified for the 4:4:4/8 profile in subclause A.3 shall be fulfilled.~~

~~Conformance of a bitstream to the 4:4:4/8 profile is specified by profile_idc being equal to 71. Decoders conforming to the 4:4:4/8 profile at a specific level shall be capable of decoding all bitstreams in which profile_idc is equal to 66 or 71 or constraint_set1_flag or constraint_set4_flag is equal to 1 and in which level_idc represents a level less than or equal to the specified level.~~

~~[Ed. Note: What about constraint_setN_flag? Creating a new one for each profile?]~~

1041036965) Subclause A.3.1 "Profile-independent level limits"

In subclause A.3.1, make the following changes.

Rename subclause A.3.1 with the following.

A.3.1 Level limits common to the Baseline, Main, and Extended profiles

In subclause A.3.1, make the following changes.

*Replace the phrase "*any profile*" with "*the Baseline, Main, or Extended profiles*".*

Change the last two paragraphs of subclause A.3.1 and the content of Table A-1 to the following:

Table A-1 below specifies the limits for each level. Entries marked "-" in Table A-1 denote the absence of a corresponding limit. For purposes of comparison of level capabilities, a level shall be considered to be a lower (higher) level than some other level if the level appears nearer to the top (bottom) row of Table A-1 than the other level.

A level to which the bitstream conforms shall be indicated by the syntax elements level_idc and constraint_set3_flag as follows.

- If level_idc is equal to 11 and constraint_set3_flag is equal to 1, the indicated level is level 1b.
- Otherwise (level_idc is not equal to 11 or constraint_set3_flag is not equal to 1), level_idc shall be set equal to a value of ten times the level number specified in Table A-1 and constraint_set3_flag shall be set equal to 0.

Table A-1 – Level limits

Level number	Max macroblock processing rate MaxMBPS (MB/s)	Max frame size MaxFS (MBs)	Max decoded picture buffer size MaxDPB	Max video bit rate MaxBR (1000 bits/s, 1200 bits/s, 2000 bits/s, or 2400 bits/s)	Max CPB size MaxCPB (1000 bits, 1200 bits, 2000 bits, or 2400 bits)	Vertical MV component range MaxVmvR (luma frame samples)	Min compression ratio MinCR	Max number of motion vectors per two consecutive MBs MaxMvsPer2Mb
1	1 485	99	148.5	64	175	[-64,+63.75]	2	-
1b	1 485	99	148.5	128	350	[-64,+63.75]	2	-
1.1	3 000	396	337.5	192	500	[-128,+127.75]	2	-
1.2	6 000	396	891.0	384	1 000	[-128,+127.75]	2	-
1.3	11 880	396	891.0	768	2 000	[-128,+127.75]	2	-
2	11 880	396	891.0	2 000	2 000	[-128,+127.75]	2	-
2.1	19 800	792	1 782.0	4 000	4 000	[-256,+255.75]	2	-
2.2	20 250	1 620	3 037.5	4 000	4 000	[-256,+255.75]	2	-
3	40 500	1 620	3 037.5	10 000	10 000	[-256,+255.75]	2	32
3.1	108 000	3 600	6 750.0	14 000	14 000	[-512,+511.75]	4	16
3.2	216 000	5 120	7 680.0	20 000	20 000	[-512,+511.75]	4	16
4	245 760	8 192	12 288.0	20 000	25 000	[-512,+511.75]	4	16
4.1	245 760	8 192	12 288.0	50 000	62 500	[-512,+511.75]	2	16
4.2	491 520	8 192	12 288.0	50 000	62 500	[-512,+511.75]	2	16
5	589 824	22 080	41 400.0	135 000	135 000	[-512,+511.75]	2	16
5.1	983 040	36 864	69 120.0	240 000	240 000	[-512,+511.75]	2	16

1051047066) Subclause A.3.2 "Profile-specific level limits"

Replace subclause A.3.2 with the following.

A.3.2 Profile-specific level limits

a) In bitstreams conforming to the Main, 4:2:0/10, 4:2:2/8, 4:2:2/10, or 4:4:4/12~~4:2:2/10, 4:4:4/12, or 4:4:4/8~~ profiles, the removal time of access unit 0 shall satisfy the constraint that the number of slices in picture 0 is less than or equal to (PicSizeInMbs + MaxMBPS * (t_r(0) - $t_{r,n}$(0))) ÷ SliceRate, where SliceRate is the value specified in Table A-3 that applies to picture 0.

b) In bitstreams conforming to the Main, 4:2:0/10, 4:2:2/8, 4:2:2/10, or 4:4:4/12~~4:2:2/10, 4:4:4/12, or 4:4:4/8~~ profiles, the difference between consecutive removal time of access units n and n - 1 (with n > 0) shall satisfy the constraint that the number of slices in picture n is less than or equal to MaxMBPS * (t_r(n) - t_r(n - 1)) ÷ SliceRate, where SliceRate is the value specified in Table A-3 that applies to picture n.

c) In bitstreams conforming to the Main, 4:2:0/10, 4:2:2/8, 4:2:2/10, or 4:4:4/12~~4:2:2/10, 4:4:4/12, or 4:4:4/8~~ profiles, sequence parameter sets shall have direct_8x8_inference_flag equal to 1 for the levels specified in Table A-3.

NOTE – direct_8x8_inference_flag is not relevant to the Baseline profile as the Baseline profile does not allow B slice types (specified in subclause A.2.1), and direct_8x8_inference_flag is equal to 1 for all levels of the Extended profile (as specified in subclause A.2.3).

d) In bitstreams conforming to the Main, 4:2:0/10, 4:2:2/8, 4:2:2/10, or 4:4:4/12~~4:2:2/10, 4:4:4/12, 4:4:4/8~~ or Extended profiles, sequence parameter sets shall have frame_mbs_only_flag equal to 1 for the levels specified in Table A-3 for the Main, 4:2:2/10, 4:4:4/12, and 4:4:4/8 profiles and in Table A-4 for the Extended profile.

NOTE – frame_mbs_only_flag is equal to 1 for all levels of the Baseline profile (specified in subclause A.2.1).

e) In bitstreams conforming to the Main, 4:2:0/10, 4:2:2/8, 4:2:2/10, or 4:4:4/12~~4:2:2/10, 4:4:4/12, 4:4:4/8~~ or Extended profiles, the value of sub_mb_type in B macroblocks shall not be equal to B_Bi_8x4, B_Bi_4x8, or B_Bi_4x4 for the levels in which MinLumaBiPredSize is shown as 8x8 in Table A-3 for the Main, 4:2:2/10, 4:4:4/12, and 4:4:4/8 profiles and in Table A-4 for the Extended profile.

f) In bitstreams conforming to the Baseline and Extended profiles, ($xInt_{max} - xInt_{min} + 6$) * ($yInt_{max} - yInt_{min} + 6$) <= MaxSubMbRectSize in macroblocks coded with mb_type equal to P_8x8, P_8x8ref0 or B_8x8 for all invocations of the process specified in subclause 8.4.2.2.1 used to generate the predicted luma sample array for a single list (list 0 or list 1) for each 8x8 sub-macroblock, where NumSubMbPart(sub_mb_type) > 1, where MaxSubMbRectSize is specified in Table A-2 for the Baseline profile and in Table A-4 for the Extended profile and

- $xInt_{min}$ as the minimum value of $xInt_L$ among all luma sample predictions for the sub-macroblock
- $xInt_{max}$ as the maximum value of $xInt_L$ among all luma sample predictions for the sub-macroblock
- $yInt_{min}$ as the minimum value of $yInt_L$ among all luma sample predictions for the sub-macroblock
- $yInt_{max}$ as the maximum value of $yInt_L$ among all luma sample predictions for the sub-macroblock

g) ~~g) [Ed. Note: fix formatting]~~ In bitstreams conforming to the 4:2:0/10 profile, the VCL HRD parameters, BitRate[SchedSelIdx] <= 1500 * MaxBR and CpbSize[SchedSelIdx] <= 1500 * MaxCPB for at least one value of SchedSelIdx, where BitRate[SchedSelIdx] is given by Equation E-13 and CpbSize[SchedSelIdx] is given by Equation E-14 when vcl_hrd_parameters_present_flag is equal to 1. MaxBR and MaxCPB are specified in Table A-1 in units of 1500 bits/s and 1500 bits, respectively. The bitstream shall satisfy these conditions for at least one value of SchedSelIdx in the range 0 to cpb_cnt_minus1, inclusive. CpbSize[SchedSelIdx] is also called CPB size.

h) ~~h) [Ed. Note: fix formatting]~~ In bitstreams conforming to the 4:2:0/10 profile, the NAL HRD parameters, BitRate[SchedSelIdx] <= 1800 * MaxBR and CpbSize[SchedSelIdx] <= 1800 * MaxCPB for at least one value of SchedSelIdx, where BitRate[SchedSelIdx] is given by Equation E-13 and CpbSize[SchedSelIdx] is given by Equation E-14 when nal_hrd_parameters_present_flag equal to 1. MaxBR and MaxCPB are specified in Table A-1 in units of 1800 bits/s and 1800 bits, respectively. The bitstream shall satisfy these conditions for at least one value of SchedSelIdx in the range 0 to cpb_cnt_minus1.

i) ~~i) [Ed. Note: fix formatting]~~ In bitstreams conforming to the 4:2:2/8 profile, the VCL HRD parameters, BitRate[SchedSelIdx] <= 3000 * MaxBR and CpbSize[SchedSelIdx] <= 3000 * MaxCPB for at least one value of SchedSelIdx, where BitRate[SchedSelIdx] is given by Equation E-13 and CpbSize[SchedSelIdx] is given by Equation E-14 when vcl_hrd_parameters_present_flag is equal to 1. MaxBR and MaxCPB are specified in Table A-1 in units of 3000 bits/s and 3000 bits, respectively. The bitstream shall satisfy these conditions for at least one value of SchedSelIdx in the range 0 to cpb_cnt_minus1, inclusive. CpbSize[SchedSelIdx] is also called CPB size.

j) ~~j) [Ed. Note: fix formatting]~~ In bitstreams conforming to the 4:2:2/8 profile, the NAL HRD parameters, BitRate[SchedSelIdx] <= 3600 * MaxBR and CpbSize[SchedSelIdx] <= 3600 * MaxCPB for at least one value of SchedSelIdx, where BitRate[SchedSelIdx] is given by Equation E-13 and CpbSize[SchedSelIdx] is given by Equation E-14 when nal_hrd_parameters_present_flag equal to 1. MaxBR and MaxCPB are specified in Table A-1 in units of 3600 bits/s and 3600 bits, respectively. The bitstream shall satisfy these conditions for at least one value of SchedSelIdx in the range 0 to cpb_cnt_minus1.

k) ~~k) [Ed. Note: fix formatting]~~ In bitstreams conforming to the 4:2:2/10 profile and the 4:4:4/12 profile, the VCL HRD parameters, BitRate[SchedSelIdx] <= 4000 * MaxBR and CpbSize[SchedSelIdx] <= 4000 * MaxCPB for at least one value of SchedSelIdx, where BitRate[SchedSelIdx] is given by Equation E-13 and CpbSize[SchedSelIdx] is given by Equation E-14 when vcl_hrd_parameters_present_flag is equal to 1. MaxBR and MaxCPB are specified in Table A-1 in units of 4000 bits/s and 4000 bits, respectively. The bitstream shall satisfy these conditions for at least one value of SchedSelIdx in the range 0 to cpb_cnt_minus1, inclusive. CpbSize[SchedSelIdx] is also called CPB size.

l) ~~l) [Ed. Note: fix formatting]~~ In bitstreams conforming to the 4:2:2/10 profile and the 4:4:4/12 profile, the NAL HRD parameters, BitRate[SchedSelIdx] <= 4800 * MaxBR and CpbSize[SchedSelIdx] <= 4800 * MaxCPB for at least one value of SchedSelIdx, where BitRate[SchedSelIdx] is given by Equation E-13 and CpbSize[SchedSelIdx] is given by Equation E-14 when nal_hrd_parameters_present_flag equal to 1. MaxBR

and MaxCPB are specified in Table A-1 in units of 4800 bits/s and 4800 bits, respectively. The bitstream shall satisfy these conditions for at least one value of SchedSelIdx in the range 0 to cpb_cnt_minus1.

1061057167) Subclause A.3.2.1 "Baseline profile limits" Table A-2

In subclause A.3.2.2, replace Table A-2 with the following.

Table A-2 – Baseline profile level limits

Level number	MaxSubMbRectSize
1	576
1b	576
1.1	576
1.2	576
1.3	576
2	576
2.1	576
2.2	576
3	576
3.1	-
3.2	-
4	-
4.1	-
4.2	-
5	-
5.1	-

1071067268) Subclause A.3.2.2 "Main profile limits"

In subclause A.3.2.2, make the following changes.

Rename the subclause as follows:

A.3.2.2 Main, 4:2:0/10, 4:2:2/8, 4:2:2/10, or 4:4:4/12~~4:2:2/10, 4:4:4/12, and 4:4:4/8~~ profile limits

*Replace the phrase "*conforming to the Main profile*" with "*conforming to the Main, 4:2:0/10, 4:2:2/8, 4:2:2/10, or 4:4:4/12~~4:2:2/10, 4:4:4/12, or 4:4:4/8~~ profiles*".*

Replace Table A-3 with the following:

Table A-3 – Main, 4:2:0/10, 4:2:2/8, 4:2:2/10, or 4:4:4/12~~4:2:2/10, 4:4:4/12, and 4:4:4/8~~ profile level limits

Level number	SliceRate	MinLumaBiPredSize	direct_8x8_inference_flag	frame_mbs_only_flag
1	-	-	-	1
1b	-	-	-	1
1.1	-	-	-	1
1.2	-	-	-	1
1.3	-	-	-	1
2	-	-	-	1
2.1	-	-	-	-
2.2	-	-	-	-
3	22	-	1	-
3.1	60	8x8	1	-
3.2	60	8x8	1	-
4	60	8x8	1	-
4.1	24	8x8	1	-
4.2	24	8x8	1	1
5	24	8x8	1	1
5.1	24	8x8	1	1

1081077369) Subclause A.3.2.2 "Extended profile limits" Table A-4

Replace Table A-4 with the following:

Table A-4 – Extended profile level limits

Level number	MaxSubMbRectSize	MinLumaBiPredSize	frame_mbs_only_flag
1	576	-	1
1b	576	-	1
1.1	576	-	1
1.2	576	-	1
1.3	576	-	1
2	576	-	1
2.1	576	-	-
2.2	576	-	-
3	576	-	-
3.1	-	8x8	-
3.2	-	8x8	-
4	-	8x8	-
4.1	-	8x8	-
4.2	-	8x8	1
5	-	8x8	1
5.1	-	8x8	1

1091087470) Subclause A.3.3 " Effect of level limits on frame rate (informative)" Table A-5

Replace Table A-5 with the following:

Table A-5 – Maximum frame rates (frames per second) for some example frame sizes

Level:					**1**	**1b**	**1.1**	**1.2**	**1.3**	**2**	**2.1**
Max frame size (macroblocks):					**99**	**99**	**396**	**396**	**396**	**396**	**792**
Max macroblocks/second:					**1 485**	**1 485**	**3 000**	**6 000**	**11 880**	**11 880**	**19 800**
Max frame size (samples):					25 344	25 344	101 376	101 376	101 376	101 376	202 752
Max samples/second:					380 160	380 160	768 000	1 536 000	3 041 280	3 041 280	5 068 800
Format	**Luma Width**	**Luma Height**	**MBs Total**	**Luma Samples**							
SQCIF	**128**	**96**	48	12 288	30.9	30.9	62.5	125.0	172.0	172.0	172.0
QCIF	**176**	**144**	99	25 344	15.0	15.0	30.3	60.6	120.0	120.0	172.0
QVGA	**320**	**240**	300	76 800	-	-	10.0	20.0	39.6	39.6	66.0
525 SIF	**352**	**240**	330	84 480	-	-	9.1	18.2	36.0	36.0	60.0
CIF	**352**	**288**	396	101 376	-	-	7.6	15.2	30.0	30.0	50.0
525 HHR	**352**	**480**	660	168 960	-	-	-	-	-	-	30.0
625 HHR	**352**	**576**	792	202 752	-	-	-	-	-	-	25.0
VGA	**640**	**480**	1 200	307 200	-	-	-	-	-	-	-
525 4SIF	**704**	**480**	1 320	337 920	-	-	-	-	-	-	-
525 SD	**720**	**480**	1 350	345 600	-	-	-	-	-	-	-
4CIF	**704**	**576**	1 584	405 504	-	-	-	-	-	-	-
625 SD	**720**	**576**	1 620	414 720	-	-	-	-	-	-	-
SVGA	**800**	**600**	1 900	486 400	-	-	-	-	-	-	-
XGA	**1024**	**768**	3 072	786 432	-	-	-	-	-	-	-
720p HD	**1280**	**720**	3 600	921 600	-	-	-	-	-	-	-
4VGA	**1280**	**960**	4 800	1 228 800	-	-	-	-	-	-	-
SXGA	**1280**	**1024**	5 120	1 310 720	-	-	-	-	-	-	-
525 16SIF	**1408**	**960**	5 280	1 351 680	-	-	-	-	-	-	-
16CIF	**1408**	**1152**	6 336	1 622 016	-	-	-	-	-	-	-
4SVGA	**1600**	**1200**	7 500	1 920 000	-	-	-	-	-	-	-
1080 HD	**1920**	**1088**	8 160	2 088 960	-	-	-	-	-	-	-
2Kx1K	**2048**	**1024**	8 192	2 097 152	-	-	-	-	-	-	-
4XGA	**2048**	**1536**	12 288	3 145 728	-	-	-	-	-	-	-
16VGA	**2560**	**1920**	19 200	4 915 200	-	-	-	-	-	-	-
3616x1536 (2.35:1)	**3616**	**1536**	21 696	5 554 176	-	-	-	-	-	-	-
3672x1536 (2.39:1)	**3680**	**1536**	22 080	5 652 480	-	-	-	-	-	-	-
4Kx2K	**4096**	**2048**	32 768	8 388 608	-	-	-	-	-	-	-
4096x2304 (16:9)	**4096**	**2304**	36 864	9 437 184	-	-	-	-	-	-	-

Table A-5 (continued) – Maximum frame rates (frames per second) for some example frame sizes

Level:					**2.2**	**3**	**3.1**	**3.2**	**4**	**4.1**	**4.2**
Max frame size (macroblocks):					**1 620**	**1 620**	**3 600**	**5 120**	**8 192**	**8 192**	**8 192**
Max macroblocks/second:					**20 250**	**40 500**	**108 000**	**216 000**	**245 760**	**245 760**	**589 824**
Max frame size (samples):					414 720	414 720	921 600	1 310 720	2 097 152	2 097 152	2 097 152
Max samples/second:					5 184 000	10 368 000	27 648 000	55 296 000	62 914 560	62 914 560	125 829 120
Format	**Luma Width**	**Luma Height**	**MBs Total**	**Luma Samples**							
SQCIF	**128**	**96**	48	12 288	172.0	172.0	172.0	172.0	172.0	172.0	172.0
QCIF	**176**	**144**	99	25 344	172.0	172.0	172.0	172.0	172.0	172.0	172.0
QVGA	**320**	**240**	300	76 800	67.5	135.0	172.0	172.0	172.0	172.0	172.0
525 SIF	**352**	**240**	330	84 480	61.4	122.7	172.0	172.0	172.0	172.0	172.0
CIF	**352**	**288**	396	101 376	51.1	102.3	172.0	172.0	172.0	172.0	172.0
525 HHR	**352**	**480**	660	168 960	30.7	61.4	163.6	172.0	172.0	172.0	172.0
625 HHR	**352**	**576**	792	202 752	25.6	51.1	136.4	172.0	172.0	172.0	172.0
VGA	**640**	**480**	1 200	307 200	16.9	33.8	90.0	172.0	172.0	172.0	172.0
525 4SIF	**704**	**480**	1 320	337 920	15.3	30.7	81.8	163.6	172.0	172.0	172.0
525 SD	**720**	**480**	1 350	345 600	15.0	30.0	80.0	160.0	172.0	172.0	172.0
4CIF	**704**	**576**	1 584	405 504	12.8	25.6	68.2	136.4	155.2	155.2	172.0
625 SD	**720**	**576**	1 620	414 720	12.5	25.0	66.7	133.3	151.7	151.7	172.0
SVGA	**800**	**600**	1 900	486 400	-	-	56.8	113.7	129.3	129.3	172.0
XGA	**1024**	**768**	3 072	786 432	-	-	35.2	70.3	80.0	80.0	160.0
720p HD	**1280**	**720**	3 600	921 600	-	-	30.0	60.0	68.3	68.3	136.5
4VGA	**1280**	**960**	4 800	1 228 800	-	-	-	45.0	51.2	51.2	102.4
SXGA	**1280**	**1024**	5 120	1 310 720	-	-	-	42.2	48.0	48.0	96.0
525 16SIF	**1408**	**960**	5 280	1 351 680	-	-	-	-	46.5	46.5	93.1
16CIF	**1408**	**1152**	6 336	1 622 016	-	-	-	-	38.8	38.8	77.6
4SVGA	**1600**	**1200**	7 500	1 920 000	-	-	-	-	32.8	32.8	65.5
1080 HD	**1920**	**1088**	8 160	2 088 960	-	-	-	-	30.1	30.1	60.2
2Kx1K	**2048**	**1024**	8 192	2 097 152	-	-	-	-	30.0	30.0	60.0
4XGA	**2048**	**1536**	12 288	3 145 728	-	-	-	-	-	-	-
16VGA	**2560**	**1920**	19 200	4 915 200	-	-	-	-	-	-	-
3616x1536 (2.35:1)	**3616**	**1536**	21 696	5 554 176	-	-	-	-	-	-	-
3672x1536 (2.39:1)	**3680**	**1536**	22 080	5 652 480	-	-	-	-	-	-	-
4Kx2K	**4096**	**2048**	32 768	8 388 608	-	-	-	-	-	-	-
4096x2304 (16:9)	**4096**	**2304**	36 864	9 437 184	-	-	-	-	-	-	-

Table A-5 (concluded) – Maximum frame rates (frames per second) for some example frame sizes

Level:					**5**	**5.1**
Max frame size (macroblocks):					**21 696**	**36 864**
Max macroblocks/second:					**589 824**	**983 040**
Max frame size (samples):					5 554 176	9 437 184
Max samples/second:					150 994 944	251 658 240
Format	**Luma Width**	**Luma Height**	**MBs Total**	**Luma Samples**		
SQCIF	**128**	**96**	48	12 288	172.0	172.0
QCIF	**176**	**144**	99	25 344	172.0	172.0
QVGA	**320**	**240**	300	76 800	172.0	172.0
525 SIF	**352**	**240**	330	84 480	172.0	172.0
CIF	**352**	**288**	396	101 376	172.0	172.0
525 HHR	**352**	**480**	660	168 960	172.0	172.0
625 HHR	**352**	**576**	792	202 752	172.0	172.0
VGA	**640**	**480**	1 200	307 200	172.0	172.0
525 4SIF	**704**	**480**	1 320	337 920	172.0	172.0
525 SD	**720**	**480**	1 350	345 600	172.0	172.0
4CIF	**704**	**576**	1 584	405 504	172.0	172.0
625 SD	**720**	**576**	1 620	414 720	172.0	172.0
SVGA	**800**	**600**	1 900	486 400	172.0	172.0
XGA	**1024**	**768**	3 072	786 432	172.0	172.0
720p HD	**1280**	**720**	3 600	921 600	163.8	172.0
4VGA	**1280**	**960**	4 800	1 228 800	122.9	172.0
SXGA	**1280**	**1024**	5 120	1 310 720	115.2	172.0
525 16SIF	**1408**	**960**	5 280	1 351 680	111.7	172.0
16CIF	**1408**	**1152**	6 336	1 622 016	93.1	155.2
4SVGA	**1600**	**1200**	7 500	1 920 000	78.6	131.1
1080 HD	**1920**	**1088**	8 160	2 088 960	72.3	120.5
2Kx1K	**2048**	**1024**	8 192	2 097 152	72.0	120.0
4XGA	**2048**	**1536**	12 288	3 145 728	48.0	80.0
16VGA	**2560**	**1920**	19 200	4 915 200	30.7	51.2
3616x1536 (2.35:1)	**3616**	**1536**	21 696	5 554 176	27.2	45.3
3672x1536 (2.39:1)	**3680**	**1536**	22 080	5 652 480	26.7	44.5
4Kx2K	**4096**	**2048**	32 768	8 388 608	-	30.0
4096x2304 (16:9)	**4096**	**2304**	36 864	9 437 184	-	26.7

1101~~097571~~) New subclause A.3.2.4 [Ed. Note: Number doesn't match below. Is it A.3.2.4 or A.3.2.2?]~~"Additional 4:2:2/10, 4:4:4/12, and 4:4:4/8 profile limits"~~" Level limits common to the 4:2:0/10, 4:2:2/8, 4:2:2/10, and 4:4:4/12 profiles"

Insert a new subclause A.3.2.4 as follows.

A.3.2.2 Level limits common to the 4:2:0/10, 4:2:2/8, 4:2:2/10, and 4:4:4/12 profiles

~~Additional 4:2:2/10, 4:4:4/12, and 4:4:4/8 profile limits~~

Let the variable fR be derived as follows.

- If picture n is a frame, fR is set equal to 1 ÷ 172.
- Otherwise (picture n is a field), fR is set equal to 1 ÷ (172 * 2).

Bitstreams conforming to the 4:2:0/10, 4:2:2/8, 4:2:2/10, or 4:4:4/12~~4:2:2/10, 4:4:4/12, or 4:4:4/8~~ profiles at a specified level shall obey the following constraints:

a) The nominal removal time of access unit n (with n > 0) from the CPB as specified in subclause C.1.2, satisfies the constraint that $t_{r,n}(n) - t_r(n - 1)$ is greater than or equal to Max(PicSizeInMbs ÷ MaxMBPS, fR), where MaxMBPS is the value specified in Table A-1 that applies to picture n, and PicSizeInMbs is the number of macroblocks in picture n.

b) The difference between consecutive output times of pictures from the DPB as specified in subclause C.2.2, satisfies the constraint that $\Delta t_{o,dpb}(n)$ >= Max(PicSizeInMbs ÷ MaxMBPS, fR), where MaxMBPS is the value specified in Table A-1 for picture n, and PicSizeInMbs is the number of macroblocks of picture n, provided that picture n is a picture that is output and is not the last picture of the bitstream that is output.

c) The sum of the NumBytesInNALunit variables for access unit 0 is less than or equal to (8 * ($BitDepth_Y$ + (ChromaFormatFactor - 1) * $BitDepth_C$) * (PicSizeInMbs + MaxMBPS * ($t_r(0) - t_{r,n}(0)$)) ÷ MinCR, where MaxMBPS and MinCR are the values specified in Table A-1 that apply to picture 0 and PicSizeInMbs is the number of macroblocks in picture 0.

d) The sum of the NumBytesInNALunit variables for access unit n (with n > 0) is less than or equal to (8 * ($BitDepth_Y$ + (ChromaFormatFactor - 1) * $BitDepth_C$)) * MaxMBPS * ($t_r(n) - t_r(n – 1)$) ÷ MinCR, where MaxMBPS and MinCR are the values specified in Table A-1 that apply to picture n. *[Ed. Note: Same MinCR requirement?]*

e) PicWidthInMbs * FrameHeightInMbs <= MaxFS, where MaxFS is specified in Table A-1

f) PicWidthInMbs <= Sqrt(MaxFS * 8)

g) FrameHeightInMbs <= Sqrt(MaxFS * 8)

h) max_dec_frame_buffering <= MaxDpbSize, where MaxDpbSize is equal to Min(2048 * MaxDPB / (PicWidthInMbs * FrameHeightInMbs * 384), 16) and MaxDPB is specified in Table A-1. max_dec_frame_buffering is also called DPB size.

~~i)For the VCL HRD parameters, BitRate[SchedSelIdx] <= 2000 * MaxBR and CpbSize[SchedSelIdx] <= 2000 * MaxCPB for at least one value of SchedSelIdx, where BitRate[SchedSelIdx] is given by Equation E-13 and CpbSize[SchedSelIdx] is given by Equation E-14 when vcl_hrd_parameters_present_flag is equal to 1. MaxBR and MaxCPB are specified in Table A-1 in units of 2000 bits/s and 2000 bits, respectively. The bitstream shall satisfy these conditions for at least one value of SchedSelIdx in the range 0 to cpb_cnt_minus1, inclusive. CpbSize[SchedSelIdx] is also called CPB size.~~

~~j)For the NAL HRD parameters, BitRate[SchedSelIdx] <= 2400 * MaxBR and CpbSize[SchedSelIdx] <= 2400 * MaxCPB for at least one value of SchedSelIdx, where BitRate[SchedSelIdx] is given by Equation E-13 and CpbSize[SchedSelIdx] is given by Equation E-14 when nal_hrd_parameters_present_flag equal to 1. MaxBR and MaxCPB are specified in Table A-1 in units of 2400 bits/s and 2400 bits, respectively. The bitstream shall satisfy these conditions for at least one value of SchedSelIdx in the range 0 to cpb_cnt_minus1.~~

~~k)~~i) Vertical motion vector component range does not exceed MaxVmvR in units of luma frame samples, where MaxVmvR is specified in Table A-1

~~l)~~j) Horizontal motion vector range does not exceed the range of -2048 to 2047.75, inclusive, in units of luma samples

~~m)~~k) Number of motion vectors per two consecutive macroblocks in decoding order (also applying to the total from the last macroblock of a slice and the first macroblock of the next slice in decoding order) does not exceed MaxMvsPer2Mb, where MaxMvsPer2Mb is specified in Table A-1. The number of motion vectors for each macroblock is value of the variable MvCnt after the completion of the intra or inter prediction process for the macroblock.

~~n)~~l) Number of bits of macroblock_layer( ) data for any macroblock is not greater than 128 + 2048 * ChromaFormatFactor. Depending on entropy_coding_mode_flag, the bits of macroblock_layer( ) data are counted as follows.

- If entropy_coding_mode_flag is equal to 0, the number of bits of macroblock_layer( ) data is given by the number of bits in the macroblock_layer( ) syntax structure for a macroblock.
- Otherwise (entropy_coding_mode_flag is equal to 1), the number of bits of macroblock_layer( ) data for a macroblock is given by the number of times read_bits(1) is called in subclauses 9.3.3.2.2 and 9.3.3.2.3 when parsing the macroblock_layer( ) associated with the macroblock.

Table A-1 below specifies the limits for each level. Entries marked "-" in Table A-1 denote the absence of a corresponding limit.

A level to which the bitstream conforms shall be indicated by the syntax element level_idc as follows.

– If level_idc is equal to 9, the indicated level is level 1b.

– Otherwise (level_idc is not equal to 9), level_idc shall be set equal to a value of ten times the level number specified in Table A-1.

~~Conformance to a particular level shall be specified by setting the syntax element level_idc equal to a value of ten times the level number specified in Table A-1.~~

1111107672) Subclause D.1 "SEI payload syntax"

Replace subclause D.1 with the following.

D.1 SEI payload syntax

sei_payload(payloadType, payloadSize) {	C	Descriptor
if(payloadType = = 0)		
buffering_period(payloadSize)	5	
else if(payloadType = = 1)		
pic_timing(payloadSize)	5	
else if(payloadType = = 2)		
pan_scan_rect(payloadSize)	5	
else if(payloadType = = 3)		
filler_payload(payloadSize)	5	
else if(payloadType = = 4)		
user_data_registered_itu_t_t35(payloadSize)	5	
else if(payloadType = = 5)		
user_data_unregistered(payloadSize)	5	
else if(payloadType = = 6)		
recovery_point(payloadSize)	5	
else if(payloadType = = 7)		
dec_ref_pic_marking_repetition(payloadSize)	5	
else if(payloadType = = 8)		
spare_pic(payloadSize)	5	
else if(payloadType = = 9)		
scene_info(payloadSize)	5	
else if(payloadType = = 10)		
sub_seq_info(payloadSize)	5	
else if(payloadType = = 11)		
sub_seq_layer_characteristics(payloadSize)	5	
else if(payloadType = = 12)		
sub_seq_characteristics(payloadSize)	5	
else if(payloadType = = 13)		
full_frame_freeze(payloadSize)	5	
else if(payloadType = = 14)		
full_frame_freeze_release(payloadSize)	5	
else if(payloadType = = 15)		
full_frame_snapshot(payloadSize)	5	
else if(payloadType = = 16)		
progressive_refinement_segment_start(payloadSize)	5	
else if(payloadType = = 17)		
progressive_refinement_segment_end(payloadSize)	5	
else if(payloadType = = 18)		
motion_constrained_slice_group_set(payloadSize)	5	
else if(payloadType = = 19)		
film_grain(payloadSize)	5	
else if(payloadType = = 20)		
display_preference_for_deblocking(payloadSize)	5	

else if(payloadType = = 21)		
stereo_video_fields(payloadSize)	5	
else		
reserved_sei_message(payloadSize)	5	
if(!byte_aligned( )) {		
bit_equal_to_one /* equal to 1 */	5	f(1)
while(!byte_aligned( ))		
bit_equal_to_zero /* equal to 0 */	5	f(1)
}		
}		

1121117773) New subclause D.1.21 "Film grain SEI message syntax"

Add a new subclause D.1.21 as follows.

D.1.21 Film grain SEI message semantics

film_grain(payloadSize) {	C	Descriptor
film_grain_cancel_flag	5	u(1)
if(!film_grain_cancel_flag) {		
model_id	5	u(2)
separate_colour_description_present_flag	5	u(1)
if(separate_colour_description_present_flag) {		
film_grain_bit_depth_luma_minus8	5	u(3)
film_grain_bit_depth_chroma_minus8	5	u(3)
film_grain_full_range_flag	5	u(1)
film_grain_colour_primaries	5	u(8)
film_grain_transfer_characteristics	5	u(8)
film_grain_matrix_coefficients	5	u(8)
}		
blending_mode_id	5	u(2)
log2_scale_factor	5	u(4)
comp0_param_present_flag	5	u(1)
comp1_param_present_flag	5	u(1)
comp2_param_present_flag	5	u(1)
if(comp0_param_presence_flag)		
read_comp_parameters(0)		
if(comp1_param_presence_flag)		
read_comp_ parameters(1)		
if(comp2_param_presence_flag)		
read_comp_ parameters(2)		
film_grain_repetition_period	5	ue(v)
}		
}		

read_comp_parameters(c) {	C	Descriptor
no_intensity_intervals_minus1[c]	5	u(8)
no_params_minus1[c]	5	u(3)
for(i = 0; i <= no_intensity_intervals_minus1[c]; i++) {		
intensity_interval_lower_bound[c][i]	5	u(8)
intensity_interval_upper_bound[c][i]	5	u(8)
for(j = 0; j <= no_params_minus1[c]; j++) {		
param[c][i][j]	5	se(v)
}		
}		
}		

[Ed. Note: Don't we ordinarily only have one syntax structure per syntax subclause?]

113~~11274~~) New subclause D.1.22 "Deblocking filter display preference SEI message syntax"

Add a new subclause D.1.22 as follows.

D.1.22 Film grain SEI message semantics

deblocking_filter_display_preference(payloadSize) {	C	Descriptor
deblocking_display_preference_cancel_flag	5	u(1)
if(!deblocking_display_preference_cancel_flag) {		
display_prior_to_deblocking_preferred_flag	5	u(1)
deblocking_display_preference_repetition_period	5	ue(v)
}		
}		

114~~1137975~~) New subclause D.1.23 "Stereo video fields SEI message syntax"

Add a new subclause D.1.23 as follows.

D.1.23 Stereo video fields SEI message semantics

stereo_video_fields(payloadSize) {	C	Descriptor
top_field_is_left_view_flag	5	u(1)
top_field_self_contained_flag	5	u(1)
bot_field_self_contained_flag	5	u(1)
}		

115~~1148076~~) New subclause D.2.21 "Film grain SEI message semantics"

Add a new subclause D.2.21 as follows.

D.2.21 Film grain SEI message semantics

This SEI message provides the decoder with a parameterised model for the film grain present in the original source material. Simulation of film grain on the decoded images for the display process not specified in this Recommendation | International Standard is optional and does not affect the decoding process specified in this Recommendation | International Standard.

film_grain_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of a previous film grain SEI message. film_grain_cancel_flag equal to 0 indicates that the SEI message does not cancel the persistence of a previous film grain SEI message and that film grain information follows.

model_id identifies the film grain simulation model as specified in Table D-5.

Table D-5 – model_id values

Value	Description
0	frequency filtering
1	auto-regression
2	reserved
3	reserved

separate_colour_description_flag equal to 1 indicates that a distinct colour space description for the film grain characteri~~sz~~sation specified in the SEI message is present in the film grain SEI message syntax. separate_colour_description_flag equal to 0 indicates that the colour description for the film grain characteri~~z~~sation specified in the SEI message is the same as for the coded video sequence as specified in subclause E.2.1.

NOTE – When separate_colour_description_flag is equal to 1, the colour space specified for the film grain characteri~~sz~~sation specified in the SEI message may differ from the colour space specified for the coded video as specified in subclause E.2.1.

film_grain_bit_depth_luma_minus8 plus 8 specifies the bit depth used for the luma component of the film grain characteri~~sz~~sation specified in the SEI message. When film_grain_bit_depth_luma_minus8 is not present, film_grain_bit_depth_luma_minus8 shall be inferred to be equal to bit_depth_luma_minus8.

film_grain_bit_depth_chroma_minus8 plus 8 specifies the bit depth used for the Cb and Cr components of the film grain character~~ization~~isation specified in the SEI message. When film_grain_bit_depth_chroma_minus8 is not present, film_grain_bit_depth_luma_minus8 shall be inferred to be equal to bit_depth_chroma_minus8.

film_grain_full_range_flag has the same semantics as specified in subclause E.2.1 for the video_full_range_flag syntax element, except as follows.

– film_grain_full_range_flag specifies the colour space of the film grain character~~ization~~isation specified in the SEI message, rather than the colour space used for the coded video sequence.

– When film_grain_full_range_flag is not present in the film_grain SEI message, the value of film_grain_full_range_flag shall be inferred to be equal to video_full_range_flag.

film_grain_colour_primaries has the same semantics as specified in subclause E.2.1 for the colour_primaries syntax element, except as follows.

– film_grain_colour_primaries specifies the colour space of the film grain character~~ization~~isation specified in the SEI message, rather than the colour space used for the coded video sequence.

– When film_grain_colour_primaries is not present in the film_grain SEI message, the value of film_grain_colour_primaries shall be inferred to be equal to colour_primaries.

film_grain_transfer_characteristics has the same semantics as specified in subclause E.2.1 for the transfer_characteristics syntax element, except as follows.

– film_grain_transfer_characteristics specifies the colour space of the film grain character~~ization~~isation specified in the SEI message, rather than the colour space used for the coded video sequence.

– When film_grain_transfer_characteristics is not present in the film_grain SEI message, the value of film_grain_transfer_characteristics shall be inferred to be equal to transfer_characteristics.

film_grain_matrix_coefficients has the same semantics as specified in subclause E.2.1 for the matrix_coefficients syntax element, except as follows.

– film_grain_matrix_coefficients specifies the colour space of the film grain character~~ization~~isation specified in the SEI message, rather than the colour space used for the coded video sequence.

– When film_grain_matrix_coefficients is not present in the film_grain SEI message, the value of film_grain_matrix_coefficients shall be inferred to be equal to matrix_coefficients.

– The values allowed for film_grain_matrix_coefficients are not constrained by the value of ChromaFormatFactor.

The chroma format factor of the film grain character~~ization~~isation specified in the film grain SEI message shall be inferred to be equal to 3 (4:4:4).

NOTE - Because the film grain generation function used by the display process is non-normative, a decoder may, if desired, down-convert the parameters for chroma in order to simulate film grain for other format factors (4:2:0 or 4:2:2) rather than up-converting the decoded video (using a method not specified by this Recommendation | International Standard) before performing film grain generation.

[Ed. Note - RGBlog and XYZ should be supported in Annex E.]

blending_mode_id identifies the blending mode used to blend the simulated film grain with the decoded images as specified in Table D-6.

Table D-6 – blending_mode_id values

Value	Description
0	additive
1	multiplicative
2	reserved
3	reserved

When blending_mode_id is equal to 0 the blending mode is additive as specified by the following equation:

$$I_{grain}[x, y, c] = Clip3(0, (1 << bitDepth[c]) – 1, I_{decoded}[x, y, c] + G[x, y, c]) \quad (D\text{-}14)$$

where $I_{decoded}[x, y, c]$ represents the sample value at coordinates [x,y] of the colour component c of the decoded image $I_{decoded}$, G[x, y, c] is the simulated film grain value at the same position and colour component, and bitDepth[c] is the number of bits used for each sample in a fixed-length unsigned binary representation of the array $I_{grain}[x, y, c]$.

Following the same notation, when blending_mode_id equals 1 the blending mode is multiplicative as specified by the following equation:

$$I_{grain}[x, y, c] = Clip3(0, (1 << bitDepth[c]) – 1, I_{decoded}[x, y, c] * (1 + G[x, y, c])) \quad (D\text{-}15)$$

log2_scale_factor specifies the scale factor that shall be used to operate with integer arithmetic.

comp0_param_present_flag equal to 0 indicates that film grain is not modelled on the first colour component according to the chroma format component order list as specified in Table 6-1. comp0_param_present_flag equal to 1 indicates specific parameters modelling the film grain on the colour component 0 are present in the SEI message.

comp1_param_present_flag equal to 0 indicates that film grain is not modelled on the second colour component according to the chroma format component order list as specified in Table 6-1. comp1_param_present_flag equal to 1 indicates specific parameters modelling the film grain on the colour component 1 are present in the SEI message.

comp2_param_present_flag equal to 0 indicates that film grain is not modelled on the third colour component according to the chroma format component order list as specified in Table 6-1. comp2_param_present_flag equal to 1 indicates specific parameters modelling the film grain on the colour component 2 are present in the SEI message.

no_intensity_intervals_minus1[c] plus 1 specifies the number of intensity intervals for which a specific set of parameters has been estimated.

NOTE - The intensity intervals may overlap in order to simulate multi-generational film grain.

no_params_minus1[c] plus 1 specifies the number of parameters present for each intensity interval in which the film grain has been modelled. The value of no_params_minus1[c] shall be in the range [0,5].

intensity_interval_lower_bound[c][i] specifies the lower bound of the interval i of intensity levels for which the set of parameters that follows applies.

intensity_interval_upper_bound[c][i] specifies the upper bound of the interval i of intensity levels for which the set of parameters that follows applies.

When mode_id is equal to 0, the average value of each block b of 16x16 samples in $I_{decoded}$, referred as b_{avg}, is used to select the sets of parameters with index s[j] that apply to all the samples in the block:

```
for( i=0, j=0; i <= no_intensity_intervals_minus1; i++ ) {
   if( b_avg >= intensity_interval_lower_bound[ c ][ i ] &&
       b_avg <= intensity_interval_upper_bound[ c ][ i ] ){                (D-16)
   s[ j ] = i
   j++
}
```

When mode_id is equal to 1, the sets of parameters used to generate the film grain are selected for each sample value in $I_{decoded}$ as follows:

```
for( i=0, j=0; i <= no_intensity_intervals_minus1; i++ ) {
   if( I_decoded[ x,y,c ] >= intensity_interval_lower_bound[ c ][ i ] &&
       I_decoded[ x,y,c ] <= intensity_interval_upper_bound[ c ][ i ] ){   (D-17)
   s[ j ] = i
   j++
}
```

Samples that do not fall into any of the defined intervals are not modified by the grain generation function. Samples that fall into more than one interval will originate multi-generation grain. Multi-generation grain results from adding the grain computed independently for each intensity interval.

param[c][i][j] represents each one of the parameters present for the colour component c and the intensity interval i. The set of parameters has different meaning depending on the mode_id value.

When mode_id equals 0, a frequency filtering model enables simulating the original film grain as follows:

```
G[ x, y, 0] = ( param[ 0 ][ s ][ 0 ] * Q[ x, y, 0] ) >> log2_scale_factor   (D-18)
```

```
G[ x, y, c] = ( param[ c ][ s ][ 0 ] * Q[ x, y, c] + param[ c ][ s ][ 5 ] * G[ x, y, c-1 ] ) >>
log2_scale_factor,   for 0<c<3                                              (D-19)
```

where Q[c] is a two-dimensional random process generated by filtering blocks b_N of 16x16 random values, generated with a normalized Gaussian distribution N(0,1). The band-pass filtering of blocks b_N can be performed in the DCT domain as follows:

```
B_N = DCT16x16(b_N)
```

```
for( y=0; y<16; y++ )
   for( x=0; x<16; x++)
      if( ( x < param[ c ][ s ][ 4 ] && y < param[ c ][ s ][ 3 ]) ||        (D-20)
          x > param[ c ][ s ][ 1 ] || y > param[ c ][ s ][ 2 ])
   B_N[ x, y ] = 0
```

```
b'_N = IDCT16x16( B_N)
```

Q[c] is formed by the filtered blocks b'_N.

NOTE - Coded parameters are based on blocks of 16x16, but decoder implementation may use other block sizes. As an example, decoders implementing the DCT on blocks of 8x8, should down-convert by a factor of two the set of coded parameters param[c][s][i] for $1 \le i \le 4$.

NOTE - To reduce the blockiness that can result from mosaicing, the frequency filtered blocks b'_N, decoders may apply a low-pass filter to the block transitions.

When mode_id equals 1, an auto-regression model enables simulating the original film grain as follows:

```
G[ x, y, c] = ( param[ c ][ s ][ 0 ] * n +
```

```
param[ c ][ s ][ 1 ] * ( G[ x-1, y, c] + param[ c ][ s ][ 4 ] * G[ x, y-1, c]  >> log2_scale_factor ) +
param[ c ][ s ][ 3 ] * ( param[ c ][ s ][ 4 ] * G[ x-1, y-1, c] >> log2_scale_factor + G[ x+1, y-1, c] ) +
param[ c ][ s ][ 5 ] * ( G[ x-2, y, c] + param[ c ][ s ][ 4 ] ^ 2 * G[ x, y-2, c] >> 2 * log2_scale_factor) +
param[ c ][ s ][ 2 ] *   G[ x, y, c-1 ] ) >> log2_scale_factor                                    (D-21)
```

where n is a random value with normalized Gaussian distribution N(0,1).

param[c][i][0] provides the first parameter of the model as specified by model_id. param[c][i][0] corresponds to the standard deviation of the Gaussian noise term in the generation functions (3) and (4).

param[c][i][1] provides the second parameter of the model as specified by model_id.

When model_id is equal to 0, param[c][i][1] indicates the horizontal high cut frequency to be used to filter the DCT of a block of 16x16 random values. param[c][i][1] shall be positive and smaller than 16. When model_id is equal to 1, param[c][i][1] indicates the first order spatial correlation for neighbouring samples (x-1,y) and (x,y-1).

When not present, param[c][i][1] shall be inferred to be equal to 8 if model_id is equal to 0, and shall be inferred to be equal to 0 if model_id is equal to 1.

param[c][i][2] provides the third parameter of the model as specified by model_id.

When model_id is equal to 0, parameter[c][i][2] indicates the vertical high cut frequency to be used to filter the DCT of a block of 16x16 random values. param[c][i][2] shall be positive and smaller than 16. When model_id is equal to 1, param[c][i][1] indicates the colour correlation between consecutive colour components.

When not present, param[c][i][2] shall be inferred to be equal to param[c][i][1] if model_id is equal to 0, and shall be inferred to be equal to 0 if model_id is equal to 1.

param[c][i][3] provides the fourth parameter of the model as specified by model_id.

When model_id is equal to 0, parameter[c][i][3] indicates the horizontal low cut frequency to be used to filter the DCT of a block of 16x16 random values. param[c][i][3] shall be positive and smaller or equal than param[c][i][1] . When model_id is equal to 1, parameter[c][i][3] indicates the first order spatial correlation for neighbouring samples (x-1,y-1) and (x+1,y-1).

When not present, param[c][i][3] shall be inferred to be equal to 0.

param[c][i][4] provides the fifth parameter of the model as specified by model_id.

When model_id is equal to 0, param[c][i][4] indicates the vertical low cut frequency to be used to to filter the DCT of a block of 16x16 random values. param[c][i][4] shall be positive and smaller or equal than param[c][i][2]. When model_id is equal to 1, param[c][i][4] indicates the aspect ratio of the modelled grain.

When not present, param[c][i][4] shall be inferred to be equal to 0 if model_id is equal to 0, and shall be inferred to be equal to 1 if model_id is equal to 1.

param[c][i][5] provides the sixth parameter of the model as specified by model_id.

When model_id is equal to 0, param[c][i][5] indicates the colour correlation between consecutive colour components. When model_id is equal to 1, param[c][i][5] indicates the second order spatial correlation for neighbouring samples (x,y-2) and (x-2,y).

When not present, param[c][i][5] shall be inferred to be equal to 0.

film_grain_repetition_period indicates whether another film grain SEI message shall be present in the bitstream and specifies the picture order count interval within which another film grain SEI message will be present. The value of film_grain_repetition_period shall be in the range 0 to 16 384, inclusive.

film_grain_repetition_period equal to 0 specifies that the film grain SEI message applies to the current decoded picture only.

film_grain_repetition_period equal to 1 specifies that the film grain SEI message persists in output order until any of the following conditions are true.

- A new coded video sequence begins
- A picture in an access unit containing a film grain SEI message that is output having PicOrderCnt( ) greater than PicOrderCnt(CurrPic).

film_grain_repetition_period greater than 1 specifies that the film grain SEI message persists until any one of the following conditions are true.

- A new coded video sequence begins
- A picture in an access unit containing a film grain SEI message is output having PicOrderCnt( ) greater than PicOrderCnt(CurrPic) + film_grain_repetition_period.

film_grain_repetition_period greater than 1 indicates that another film grain SEI message shall be present for a picture in an access unit that is output having PicOrderCnt( ) less than or equal to PicOrderCnt(CurrPic) + film_grain_repetition_period; unless a new coded video sequence begins without output of such a picture.

1161158177) New subclause D.2.22 "Deblocking filter display preference SEI message semantics"

Add a new subclause D.2.22 as follows.

D.2.22 Deblocking filter display preference SEI message semantics

This SEI message provides the decoder with an indication of whether the display of the cropped output of the deblocking filter process specified in subclause 8.7 or of the cropped output of the picture construction process prior to the deblocking filter process specified in subclause 8.5.9 is preferred by the encoder for the display of each decoded picture that is output.

NOTE – The display process is not specified in this Recommendation | International Standard. The means by which an encoder determines what to indicate as its preference expressed in a deblocking filter display preference SEI message is also not specified in this Recommendation | International Standard, and the expression of an expressed preference in a deblocking filter display preference SEI message does not impose any requirement on the display process.

deblocking_display_preference_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of a previous deblocking filter display preference SEI message. deblocking_display_preference_cancel_flag equal to 0 indicates that the SEI message does not cancel the persistence of a previous deblocking filter display preference SEI message and that a display_prior_to_deblocking_preferred_flag and deblocking_display_preference_repetition_period follow.

NOTE – In the absence of the deblocking filter display preference SEI message, or after the receipt of a deblocking filter display preference SEI message in which deblocking_filter_display_preference_cancel_flag is equal to 1, the decoder should infer that the display of the cropped output of the deblocking filter process specified in subclause 8.7 is preferred over the display of the cropped output of the picture construction process prior to the deblocking filter process specified in subclause 8.5.9 for the display of each decoded picture that is output.

display_prior_to_deblocking_preferred_flag equal to 1 indicates that the encoder preference is for the display process (which is not specified in this Recommendation | International Standard) to display the cropped output of the picture construction process prior to the deblocking filter process specified in subclause 8.5.9 rather than the cropped output of the deblocking filter process specified in subclause 8.7 for each picture that is cropped and output as specified in Annex C. display_prior_to_deblocking_preferred_flag equal to 0 indicates that the encoder preference is for the display process (which is not specified in this Recommendation | International Standard) to display the cropped output of the deblocking filter process specified in subclause 8.7 rather than the cropped output of the picture construction process prior to the deblocking filter process specified in subclause 8.5.9 for each picture that is cropped and output as specified in Annex C.

NOTE – The presence or absence of the deblocking filter display preference SEI message and the value of display_prior_to_deblocking_preferred_flag does not affect the requirements of the decoding process specified in this Recommendation | International Standard. Rather, it only provides an indication of when, in addition to fulfilling the requirements of this Recommendation | International Standard for the decoding process, enhanced visual quality may be obtained by performing the display process (which is not specified in this Recommendation | International Standard) in an alternative fashion. Decoders designed to take advantage of the content of the deblocking filter display preference SEI message should have significant added capacity for the storage of the output of the picture construction process prior to the deblocking filter process specified in subclause 8.5.9 in addition to the storage of the output of the deblocking filter process specified in subclause 8.7 when reordering and delaying pictures for display.

deblocking_display_preference_repetition_period indicates whether another deblocking filter display preference SEI message shall be present in the bitstream and specifies the picture order count interval within which another deblocking filter display preference SEI message will be present. The value of deblocking_display_preference_repetition_period shall be in the range 0 to 16 384, inclusive.

deblocking_display_preference_repetition_period equal to 0 specifies that the deblocking filter display preference SEI message applies to the current decoded picture only.

deblocking_display_preference_repetition_period equal to 1 specifies that the deblocking filter display preference SEI message persists in output order until any of the following conditions are true.

– A new coded video sequence begins

– A picture in an access unit containing a deblocking filter display preference SEI message that is output having PicOrderCnt( ) greater than PicOrderCnt(CurrPic).

deblocking_display_preference_repetition_period greater than 1 specifies that the deblocking filter display preference SEI message persists until any one of the following conditions are true.

– A new coded video sequence begins

– A picture in an access unit containing a deblocking filter display preference SEI message is output having PicOrderCnt( ) greater than PicOrderCnt(CurrPic) + deblocking_display_preference_repetition_period.

deblocking_display_preference_repetition_period greater than 1 indicates that another deblocking filter display preference SEI message shall be present for a picture in an access unit that is output having PicOrderCnt( ) less than or equal to PicOrderCnt(CurrPic) + deblocking_display_preference_repetition_period; unless a new coded video sequence begins without output of such a picture.

1171~~16827~~8) New subclause D.2.23 "Stereo video fields SEI message semantics"

Add a new subclause D.2.23 as follows.

D.2.23 Stereo video fields SEI message semantics

This SEI message provides the decoder with an indication that the entire coded video sequence consists of a sequence of coded fields in which all fields of a particular parity represent a left view and all fields of the opposite parity represent a right view for stereo-view video content.

The stereo video fields SEI message shall not be present unless all pictures in the coded video sequence are fields.

The stereo video fields SEI message shall not be present in any access unit of a coded video sequence unless a stereo video fields SEI message is present in the first access unit of the coded video sequence. The stereo video fields SEI message may also be present in other access units of the coded video sequence in addition to being present in the first access unit of the coded video sequence.

Within a coded video sequence, the values of the syntax elements in all stereo video fields SEI messages shall be the same.

When the stereo video fields SEI message is present, the spatial locations of the samples in each individual field should be interpreted for display purposes as representing complete pictures as shown in Figure 6-1 rather than as spatially-distinct fields within a source frame as shown in Figure 6-2.

NOTE – The display process is not specified in this Recommendation | International Standard.

top_field_is_left_view_flag equal to 1 indicates that the top fields in the coded video sequence represent a view from the left from the perspective of the viewer and the bottom fields in the coded video sequence represent a view from the right from the perspective of the viewer. top_field_is_left_view_flag equal to 0 indicates that the bottom fields in the coded video sequence represent a view from the left from the perspective of the viewer and the top fields in the coded video sequence represent a view from the right from the perspective of the viewer.

top_field_self_contained_flag equal to 1 indicates that no inter prediction operations within the decoding process for the top fields of the coded video sequence refer to reference pictures that are bottom fields.
top_field_self_contained_flag equal to 0 indicates that some inter prediction operations within the decoding process for the top fields of the coded video sequence may or may not refer to reference pictures that are bottom fields.

bot_field_self_contained_flag equal to 1 indicates that no inter prediction operations within the decoding process for the bottom fields of the coded video sequence refer to reference pictures that are top fields.
bot_field_self_contained_flag equal to 0 indicates that some inter prediction operations within the decoding process for the bottom fields of the coded video sequence may or may not refer to reference pictures that are top fields.

[Ed. Note: What does this SEI message say about the display timing?]

1181178379) Subclause E.2.1 "VUI parameters semantics"

Replace the section starting with "video_full_range_flag indicates" and ending with "shall be inferred to be equal to 0" with the following.

video_full_range_flag indicates the black level and range of the luma and chroma signals as derived from E'_Y, E'_{PB}, and E'_{PR} or E'_R, E'_G, and E'_B analogue component signals.

When the video_full_range_flag syntax element is not present, video_full_range_flag value shall be inferred to be equal to 0.

Replace the section starting with "matrix_coefficients describes" and ending with "shall be inferred to be equal to 2" with the following.

matrix_coefficients describes the matrix coefficients used in deriving luma and chroma signals from the green, blue, and red primaries, as specified in Table E-5.

matrix_coefficients shall not be equal to 0 unless both of the following conditions are true

– $BitDepth_C$ is equal to $BitDepth_Y$

– ChromaFormatFactor is equal to 3 (4:4:4)

matrix_coefficients shall not be equal to 8 unless both of the following conditions are true

– $BitDepth_C$ is equal to $BitDepth_Y + 1$

– ChromaFormatFactor is equal to 3 (4:4:4)

The interpretation of matrix_coefficients is defined as follows.

– E'_R, E'_G, and E'_B are analogue with values in the range of 0 to 1.
– White is specified as having E'_R equal to 1, E'_G equal to 1, and E'_B equal to 1.
– Black is specified as having E'_R equal to 0, E'_G equal to 0, and E'_B equal to 0.
– If video_full_range_flag is equal to 0, the following equations apply.
  – If matrix_coefficients is equal to 1, 4, 5, 6, or 7, the following equations apply.

$$Y = Round((1 << (BitDepth_Y - 8)) * (219 * E'_Y + 16)) \quad \text{(E-1Y)}$$

$$Cb = Round((1 << (BitDepth_C - 8)) * (224 * E'_{PB} + 128)) \quad \text{(E-2Cb)}$$

$$Cr = Round((1 << (BitDepth_C - 8)) * (224 * E'_{PR} + 128)) \quad \text{(E-3Cr)}$$

  – Otherwise (matrix_coefficients is not equal to 1, 4, 5, 6, or 7), if matrix_coefficients is equal to 0 or 8, the following equations apply.

$$R = Round((1 << (BitDepth_Y - 8)) * (219 * E'_R + 16)) \quad \text{(E-1R)}$$

$$G = Round((1 << (BitDepth_Y - 8)) * (219 * E'_G + 16)) \quad \text{(E-2G)}$$

$$B = Round((1 << (BitDepth_Y - 8)) * (219 * E'_B + 16)) \quad \text{(E-3B)}$$

  – Otherwise (matrix_coefficients is not equal to 0, 1, 4, 5, 6, 7, or 8), if matrix_coefficients is equal to 2, the interpretation of the matrix_coefficients syntax element is unknown or is determined by the application.
  – Otherwise (matrix_coefficients is not equal to 0, 1, 2, 4, 5, 6, 7, or 8), the interpretation of the matrix_coefficients syntax element is reserved for future definition by ITU-T | ISO/IEC.
– Otherwise (video_full_range_flag is equal to 1), the following equations apply.
  – If matrix_coefficients is equal to 1, 4, 5, 6, or 7, the following equations apply.

$$Y = Round(((1 << BitDepth_Y) - 1) * E'_Y) \quad \text{(E-4Y)}$$

$$Cb = Round(((1 << BitDepth_C) – 1) * E'_{PB} + (1 << (BitDepth_C – 1)) \quad \text{(E-5Cb)}$$

$$Cr = Round(((1 << BitDepth_C) – 1) * E'_{PR} + (1 << (BitDepth_C – 1)) \quad \text{(E-6Cr)}$$

– Otherwise (matrix_coefficients is not equal to 1, 4, 5, 6, or 7), if matrix_coefficients is equal to 0 or 8, the following equations apply.

$$R = Round(((1 << BitDepth_Y) – 1) * E'_R) \quad \text{(E-4R)}$$

$$G = Round(((1 << BitDepth_Y) – 1) * E'_G) \quad \text{(E-5G)}$$

$$B = Round(((1 << BitDepth_Y) – 1) * E'_B) \quad \text{(E-6B)}$$

– Otherwise (matrix_coefficients is not equal to 0, 1, 4, 5, 6, 7, or 8), if matrix_coefficients is equal to 2, the interpretation of the matrix_coefficients syntax element is unknown or is determined by the application.

– Otherwise (matrix_coefficients is not equal to 0, 1, 2, 4, 5, 6, 7, or 8), the interpretation of the matrix_coefficients syntax element is reserved for future definition by ITU-T | ISO/IEC.

– If matrix_coefficients is not equal to 0 or 8, the following equations apply.

$$E'_Y = K_R * E'_R + (1 – K_R – K_B) * E'_G + K_B * E'_B \quad \text{(E-7Y)}$$

$$E'_{PB} = 0.5 * (E'_B – E'_Y) \div (1 – K_B) \quad \text{(E-8PB)}$$

$$E'_{PR} = 0.5 * (E'_R – E'_Y) \div (1 – K_R) \quad \text{(E-9PR)}$$

NOTE – Then E'_Y is analogue with values in the range of 0 to 1, E'_{PB} and E'_{PR} are analogue with values in the range of -0.5 to 0.5, and white is equivalently given by $E'_Y = 1$, $E'_{PB} = 0$, $E'_{PR} = 0$.

– Otherwise, if matrix_coefficients is equal to 0, the following equations apply.

$$Y = G \quad \text{(E-7G)}$$

$$Cb = B \quad \text{(E-8B)}$$

$$Cr = R \quad \text{(E-9R)}$$

– Otherwise (matrix_coefficients is equal to 8), the following equations apply.

$$Cr = R – B \quad \text{(E-7Cr)}$$

$$t = B + (Cr >> 1) \quad \text{(E-8t)}$$

$$Cb = G – t \quad \text{(E-8Cb)}$$

$$Y = t + (Cb >> 1) \quad \text{(E-9Y)}$$

NOTE – The inverse conversion for the above four equations should be computed as follows.

$$t = Y – (Cb >> 1) \quad \text{(E-7t)}$$

$$G = t + Cb \quad \text{(E-7G)}$$

$$B = t - (Cr >> 1) \quad \text{(E-8B)}$$

$$R = B + Cr \quad \text{(E-9R)}$$

Table E-5 – Matrix coefficients

Value	Matrix
0	RGB
1	ITU-R Recommendation BT.709 $K_R = 0.2126$; $K_B = 0.0722$
2	Unspecified Image characteristics are unknown or are determined by the application.
3	Reserved
4	Federal Communications Commission $K_R = 0.30$; $K_B = 0.11$
5	ITU-R Recommendation BT.470-2 System B, G: $K_R = 0.299$; $K_B = 0.114$
6	Society of Motion Picture and Television Engineers 170M $K_R = 0.299$; $K_B = 0.114$
7	Society of Motion Picture and Television Engineers 240M (1987) $K_R = 0.212$; $K_B = 0.087$
8	YCoCg
9-255	Reserved

When the matrix_coefficients syntax element is not present, the value of matrix_coefficients shall be inferred to be equal to 2.

~~8480) Subclause 7.3.2.2 Picture parameter set RBSP syntax~~

~~*Replace the corresponding part of the syntax table with the following.*~~

~~entropy_coding_mode_flag~~	~~0~~	~~u(1)~~
~~if ((profile_idc == ***69 || profile_idc == 70 || profile_idc == 71 || profile_idc == 72) && entropy_coding_mode_flag) {~~		
~~**transform_8x8_mode_flag**~~	~~0~~	~~u(1)~~
~~}~~		

~~*** is the profile_idc assigned to the Fidelity Range Extensions.~~

8581) Subclause 7.3.5 Macroblock layer syntax

Replace the syntax table with the following.

macroblock_layer( ) {	C	Descriptor
mb_type	2	ue(v) | ae(v)
if(mb_type = = I_PCM) {		
while(!byte_aligned())		
pcm_alignment_zero_bit	2	f(1)
for(i = 0; i < 256 * ChromaFormatFactor; i++)		
pcm_byte[i]	2	u(8)
} else {		
NoMbPartLessThan8x8Flag = 1 /* Maybe modified in sub_mb_pred()		
if(MbPartPredMode(mb_type, 0) != Intra_NxN && MbPartPredMode(mb_type, 0) != Intra_16x16 && NumMbPart(mb_type) = = 4) {		
sub_mb_pred(mb_type)	2	
if(NoMbPartLessThan8x8Flag && transform_8x8_mode_flag)		
transform_size_flag	2	ae(v)
} else {		
if(MbPartPredMode(mb_type, 0) != Intra_16x16 && transform_8x8_mode_flag)		
transform_size_flag	2	ae(v)
mb_pred(mb_type)	2	
}		
if(MbPartPredMode(mb_type, 0) != Intra_16x16)		
coded_block_pattern	2	me(v) | ae(v)
if(CodedBlockPatternLuma > 0 || CodedBlockPatternChroma > 0 || MbPartPredMode(mb_type, 0) = = Intra_16x16) {		
mb_qp_delta	2	se(v) | ae(v)
residual( )	3 | 4	
}		
}		
}		

8682) Subclause 7.3.5.1 Macroblock prediction syntax

Replace the corresponding part of the syntax table with the following.

if(MbPartPredMode(mb_type, 0) = = Intra_4x4 || MbPartPredMode(mb_type, 0) = = Intra_8x8 || MbPartPredMode(mb_type, 0) = = Intra_16x16) {		
if(MbPartPredMode(mb_type, 0) = = Intra_4x4)		
for(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]**	2	u(1) | ae(v)
if(!prev_intra4x4_pred_mode_flag**[** luma4x4BlkIdx **]**)		
rem_intra4x4_pred_mode[luma4x4BlkIdx **]**	2	u(3) | ae(v)
}		
if(MbPartPredMode(mb_type, 0) = = Intra_8x8)		
for(luma8x8BlkIdx=0; luma8x8BlkIdx<4; luma8x8BlkIdx++) {		
prev_intra8x8_pred_mode_flag[luma8x8BlkIdx **]**	2	ae(v)
if(!prev_intra8x8_pred_mode_flag[luma8x8BlkIdx])		
rem_intra8x8_pred_mode[luma8x8BlkIdx **]**	2	ae(v)
}		

8783) Subclause 7.3.5.2 Sub-macroblock prediction syntax

Replace the syntax table with the following.

sub_mb_pred(mb_type) {	C	Descriptor
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
sub_mb_type[mbPartIdx]	2	ue(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8)		
if (NumSubMbPart(sub_mb_type[mbPartIdx]) > 1)		
NoMbPartLessThan8x8Flag = 0		
else		
if (!direct_8x8_inference_flag)		
NoMbPartLessThan8x8Flag = 0		
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if((num_ref_idx_l0_active_minus1 > 0 || mb_field_decoding_flag) && mb_type != P_8x8ref0 && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
ref_idx_l0[mbPartIdx]	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
ref_idx_l1[mbPartIdx]	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx][subMbPartIdx][compIdx]	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx][subMbPartIdx][compIdx]	2	se(v) | ae(v)
}		

8884) Subclause 7.3.5.3 Residual data syntax

Replace the corresponding part of the syntax table with the following.

for(i8x8 = 0; i8x8 < 4; i8x8++) /* each luma 8x8 block */		
if(!transform_8x8_mode_flag || !transform_size_flag || !NoMbPartLessThan8x8Flag || MbPartPredMode(mb_type, 0) = = Intra_16x16)		
for(i4x4 = 0; i4x4 < 4; i4x4++) /* each 4x4 sub-block of block */		
if(CodedBlockPatternLuma & (1 << i8x8)) {		
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
residual_block(Intra16x16ACLevel[i8x8 * 4 + i4x4], 15)	3	
else		
residual_block(LumaLevel[i8x8 * 4 + i4x4], 16)	3 | 4	
} else {		
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
for(i = 0; i < 15; i++)		
Intra16x16ACLevel[i8x8 * 4 + i4x4][i] = 0		
else		
for(i = 0; i < 16; i++)		
LumaLevel[i8x8 * 4 + i4x4][i] = 0		
}		
else		
if(CodedBlockPatternLuma & (1 << i8x8))		
residual_block(LumaLevel64[i8x8], 64)	3 | 4	
else		
for(i = 0; i < 64; i++)		
LumaLevel64[i8x8][i] = 0		

8985) Subclause 7.3.5.4 Residual block CABAC syntax

Replace the syntax table with the following.

residual_block_cabac(coeffLevel, maxNumCoeff) {	C	Descriptor
if (maxNumCoeff != 64)		
coded_block_flag /* indicates per 4x4 block whether non-zero coeffs are present */	3 | 4	ae(v)
else		
coded_block_flag = 1 /* for 8x8 blocks, presence of non-zero coeffs is implied */		
if(coded_block_flag) {		
numCoeff = maxNumCoeff		
i = 0		
do {		
significant_coeff_flag[i]	3 | 4	ae(v)
if(significant_coeff_flag[i]) {		
last_significant_coeff_flag[i]	3 | 4	ae(v)
if(last_significant_coeff_flag[i]) {		
numCoeff = i + 1		
for(j = numCoeff; j < maxNumCoeff; j++)		
coeffLevel[j] = 0		
}		
}		
i++		
} while(i < numCoeff - 1)		
coeff_abs_level_minus1[numCoeff - 1]	3 | 4	ae(v)
coeff_sign_flag[numCoeff - 1]	3 | 4	ae(v)
coeffLevel[numCoeff - 1] = (coeff_abs_level_minus1[numCoeff - 1] + 1) * (1 - 2 * coeff_sign_flag[numCoeff - 1])		
for(i = numCoeff - 2; i >= 0; i -) {		
if(significant_coeff_flag[i]) {		
coeff_abs_level_minus1[i]	3 | 4	ae(v)
coeff_sign_flag[i]	3 | 4	ae(v)
coeffLevel[i] = (coeff_abs_level_minus1[i] + 1) * (1 - 2 * coeff_sign_flag[i])		
} else		
coeffLevel[i] = 0		
}		
} else		
for(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
}		

9086) Subclause 7.4.2.2 Picture parameter set RBSP semantics

Add the following.

transform_8x8_mode_flag equal to 1 indicates that the 8x8 transform decoding process may be in use (see subclause 8.4). transform_8x8_mode_flag equal to 0 indicates that the 8x8 transform decoding process is not in use. If transform_8x8_mode_flag is not present in the bitstream, it shall be inferred to be 0.

9187) Subclause 7.4.2.2 Picture parameter set RBSP semantics

Change the corresponding parts of the subclause with the following.

Add the following paragraph.

transform_size_flag equal to 0 specifies that for the current macroblock the transform coefficient decoding process and picture construction process prior to deblocking filter process for residual 4x4 blocks shall be invoked for luma samples. transform_size_flag equal to 1 specifies that for the current macroblock the transform coefficient decoding process and picture construction process prior to deblocking filter process for residual 8x8 blocks shall be invoked for luma samples. If transform_size_flag is not present in the bitstream, it shall be inferred to be equal to 0.

The variable TransformSizeIs8x8Flag is specified as follows.

- If mb_type of the current macroblock is not equal to Intra_16x16 and transform_size_flag is equal to 1 and transform_8x8_mode_flag is equal to 1, the variable TransformSizeIs8x8Flag is set equal to 1.
- Otherwise, variable TransformSizeIs8x8Flag is set equal to 0.

Change the following parts.

Table 7-8 – Macroblock types for I slices

mb_type	Name of mb_type	transform_size_flag	MbPartPredMode (mb_type, 0)	Intra16x16PredMode	CodedBlockPatternChroma	CodedBlockPatternLuma
0	I_NxN	0	Intra_4x4	na	na	na
0	I_NxN	1	Intra_8x8	na	na	na
1	I_16x16_0_0_0	na	Intra_16x16	0	0	0
2	I_16x16_1_0_0	na	Intra_16x16	1	0	0
3	I_16x16_2_0_0	na	Intra_16x16	2	0	0
4	I_16x16_3_0_0	na	Intra_16x16	3	0	0
5	I_16x16_0_1_0	na	Intra_16x16	0	1	0
6	I_16x16_1_1_0	na	Intra_16x16	1	1	0
7	I_16x16_2_1_0	na	Intra_16x16	2	1	0
8	I_16x16_3_1_0	na	Intra_16x16	3	1	0

9	I_16x16_0_2_0	na	Intra_16x16	0	2	0
10	I_16x16_1_2_0	na	Intra_16x16	1	2	0
11	I_16x16_2_2_0	na	Intra_16x16	2	2	0
12	I_16x16_3_2_0	na	Intra_16x16	3	2	0
13	I_16x16_0_0_1	na	Intra_16x16	0	0	15
14	I_16x16_1_0_1	na	Intra_16x16	1	0	15
15	I_16x16_2_0_1	na	Intra_16x16	2	0	15
16	I_16x16_3_0_1	na	Intra_16x16	3	0	15
17	I_16x16_0_1_1	na	Intra_16x16	0	1	15
18	I_16x16_1_1_1	na	Intra_16x16	1	1	15
19	I_16x16_2_1_1	na	Intra_16x16	2	1	15
20	I_16x16_3_1_1	na	Intra_16x16	3	1	15
21	I_16x16_0_2_1	na	Intra_16x16	0	2	15
22	I_16x16_1_2_1	na	Intra_16x16	1	2	15
23	I_16x16_2_2_1	na	Intra_16x16	2	2	15
24	I_16x16_3_2_1	na	Intra_16x16	3	2	15
25	I_PCM	na	na	na	na	na

I_4x4: the macroblock is coded as an Intra_4x4 prediction macroblock I_NxN: A mnemonic name for mb_type equal to 0 with MbPartPredMode(mb_type, 0) equal to Intra_4x4 or Intra_8x8.

Intra_4x4 specifies the macroblock prediction mode and specifies that the Intra_4x4 prediction process is invoked as specified in subclause 8.3.1. Intra_4x4 is an Intra macroblock prediction mode.

Intra_8x8 specifies the macroblock prediction mode and specifies that the Intra_8x8 prediction process is invoked as specified in subclause 8.3.2. Intra_8x8 is an Intra macroblock prediction mode.

9288) Subclause 7.4.5.1 Macroblock prediction semantics

Add the following paragraph.

prev_intra8x8_pred_mode_flag[luma8x8BlkIdx] and **rem_intra8x8_pred_mode**[luma8x8BlkIdx] specify the Intra_8x8 prediction of the 8x8 luma block with index luma8x8BlkIdx = 0..3.

9389) Subclause 7.4.5.2 Sub-macroblock prediction semantics

Add the following paragraph.

The variable NoMbPartLessThan8x8Flag indicates whether for each of the 4 8x8 blocks indexed by mbPartIdx = 0..3 the corresponding SubMbPartWidth(sub_mb_type[mbPartIdx])and SubMbPartHeight(sub_mb_type[mbPartIdx]) are both equal to 8.

9490) Subclause 7.4.5.2 Sub-macroblock prediction semantics

Change the following paragraph.

Depending on MbPartPredMode(mb_type, 0), the following applies.

- If MbPartPredMode(mb_type, 0) is equal to Intra_16x16, the transform coefficient levels are parsed into the list Intra16x16DCLevel and into the 16 lists Intra16x16ACLevel[i]. Intra16x16DCLevel contains the 16 transform coefficient levels of the DC transform coefficient levels for each 4x4 luma block. For each of the 16 4x4 luma blocks indexed by i = 0..15, the 15 AC transform coefficients levels of the i-th block are parsed into the i-th list Intra16x16ACLevel[i].
- Otherwise (MbPartPredMode(mb_type, 0) is not equal to Intra_16x16), the following applies.
  - If transform_size_flag is equal to 0 or NoMbPartLessThan8x8Flag is equal to 0 or transform_8x8_mode_flag is equal to 0, for each of the 16 4x4 luma blocks indexed by i = 0..15, the 16 transform coefficient levels of the i-th block are parsed into the i-th list LumaLevel[i].
  - Otherwise (transform_size_flag is equal to 1 and NoMbPartLessThan8x8Flag is equal to 1 and transform_8x8_mode_flag is equal to 1), for each of the 4 8x8 luma blocks indexed by i = 0..3, the 64 transform coefficient levels of the i-th block are parsed into the i-th list LumaLevel64[i].

9591) Subclause 7.4.5.3.1 Residual block CABAC semantics

Change the following paragraph.

coded_block_flag specifies for blocks that are not 8x8 luma blocks whether the block contains non-zero transform coefficient levels as follows.

9692) Subclause 8.3 Intra prediction process

Change the following paragraph.

Inputs to this process are constructed samples prior to the deblocking filter process from neighbouring macroblocks and for Intra_NxN (where NxN is equal to 4x4 or 8x8) prediction mode, the associated values of IntraNxNPredMode from neighbouring macroblocks.

Outputs of this process are specified as follows.

- If mb_type is not equal to I_PCM, the Intra prediction samples of components of the macroblock or in case of the Intra_NxN prediction process for luma samples, the outputs are NxN luma sample arrays as part of the 16x16 luma array of prediction samples of the macroblock.
- Otherwise (mb_type is equal to I_PCM), constructed macroblock samples prior to the deblocking filter process.

Depending on the value of mb_type the following applies.

- If mb_type is equal to I_PCM, the process specified in subclause 8.3.5 is invoked.
- Otherwise (mb_type is not equal to I_PCM), the following applies.
  - The decoding processes for Intra prediction modes are described for the luma component as follows.
  - If the macroblock prediction mode is equal to Intra_4x4, the specification in subclause 8.3.1 applies.
  - Otherwise, if the macroblock prediction mode is equal to Intra_8x8, the specification in subclause 8.3.2 [Ed. Note (TW): new subclause] applies.
  - Otherwise (the macroblock prediction mode is equal to Intra_16x16), the specification in subclause 8.3.3 [Ed. Note (TW): incremented because of new subclause] applies.

The decoding processes for Intra prediction modes for the chroma components are described in subclause 8.3.4. [Ed. Note (TW): incremented because of new subclause]

Samples used in the Intra prediction process shall be sample values prior to alteration by any deblocking filter operations.

9793) Subclause 8.3.1 Intra_4x4 prediction process for luma samples

Change the following paragraph.

Inputs to this process are constructed luma samples prior to the deblocking filter process from neighbouring macroblocks and the associated values of Intra4x4PredMode or Intra8x8PredMode from the neighbouring macroblocks or macroblock pairs.

9894) Subclause 8.3.1.1 Derivation process for the Intra4x4PredMode

Change the following paragraph.

Inputs to this process are the index of the 4x4 luma block luma4x4BlkIdx, the index of the 8x8 luma block luma8x8BlkIdx, and variable arrays Intra4x4PredMode and Intra8x8PredMode that are previously (in decoding order) derived for adjacent macroblocks.

Change the following paragraph.

Let intraMxMPredModeA and intraMxMPredModeB be variables that specify the intra prediction modes of neighbouring 4x4 or 8x8 luma blocks. [Ed. Note (TW): improve clarity]

Intra4x4PredMode[luma4x4BlkIdx] is derived as follows.

- The process specified in subclause 6.4.7.3 is invoked with luma4x4BlkIdx given as input and the output is assigned to mbAddrA, luma4x4BlkIdxA, mbAddrB, and luma4x4BlkIdxB.
- The process specified in subclause 6.4.7.2 is invoked with luma8x8BlkIdx given as input and the output is assigned to mbAddrA, luma8x8BlkIdxA, mbAddrB, and luma8x8BlkIdxB.
- The variable dcOnlyPredictionFlag is derived as follows.

...

- For N being either replaced by A or B, the variables intraMxMPredModeN are derived as follows.
  - If dcOnlyPredictionFlag is equal to 1 or the macroblock with address mbAddrN is not coded in Intra_4x4 or Intra_8x8 macroblock prediction mode, intraMxMPredModeN is set equal to 2 (Intra_4x4_DC prediction mode).
  - Otherwise (dcOnlyPredictionFlag is equal to 0 and the macroblock with address mbAddrN is coded in Intra_4x4 or Intra_8x8 macroblock prediction mode), the following applies.
    - If the macroblock with address mbAddrN is coded in Intra_4x4 macroblock mode, intraMxMPredModeN is set equal to Intra4x4PredMode[luma4x4BlkIdxN], where Intra4x4PredMode is the variable array assigned to the macroblock mbAddrN.
    - Otherwise (the macroblock with address mbAddrN is coded in Intra_8x8 macroblock mode), intraMxMPredModeN is set equal to Intra8x8PredMode[luma8x8BlkIdxN], where Intra8x8PredMode is the variable array assigned to the macroblock mbAddrN.
- Intra4x4PredMode[luma4x4BlkIdx] is derived by applying the following procedure.

predIntra4x4PredMode = Min(intraMxMPredModeA, intraMxMPredModeB)

9995) New subclause 8.3.2 "Intra_8x8 prediction process for luma samples"

Add a new subclause 8.3.2 as follows.

This process is invoked when the macroblock prediction mode is equal to Intra_8x8.
Inputs to this process are constructed luma samples prior to the deblocking filter process from neighbouring macroblocks and the associated values of Intra4x4PredMode or Intra8x8PredMode from the neighbouring macroblocks or macroblock pairs.

Outputs of this process are 8x8 luma sample arrays as part of the 16x16 luma array of prediction samples of the macroblock $pred_L$.

The luma component of a macroblock consists of 4 blocks of 8x8 luma samples. These blocks are inverse scanned using the 8x8 luma block inverse scanning process as specified in subclause 6.4.2.

For all the 8x8 luma blocks of the luma component of a macroblock with luma8x8BlkIdx = 0..3, the variable Intra8x8PredMode[luma8x8BlkIdx] is derived as specified in subclause 8.3.2.1.

For the each luma block of 8x8 samples indexed using luma8x8BlkIdx = 0..3,

1. The Intra_8x8 sample prediction process in subclause 8.3.2.2 is invoked with luma8x8BlkIdx and constructed samples prior (in decoding order) to the deblocking filter process from adjacent luma blocks as the input and the output are the Intra_8x8 luma prediction samples $pred8x8_L[x, y]$ with x, y = 0..7.
2. The position of the upper left sample of a 8x8 luma block with index luma8x8BlkIdx inside the current macroblock is derived by invoking the inverse 8x8 luma block scanning process in subclause 6.4.2 with luma8x8BlkIdx as the input and the output being assigned to (xO, yO) and x, y = 0..7.

$$pred_L[xO + x, yO + y] = pred8x8_L[x, y]$$

3. The transform coefficient decoding process and picture construction process prior to deblocking filter process in subclause 8.5 is invoked with $pred_L$ and luma8x8BlkIdx as the input and the constructed samples for the current 8x8 luma block S'_L as the output.

10096) New subclause 8.3.2.1 "Derivation process for the Intra8x8PredMode"

Add a new subclause 8.3.2.1 as follows.

Inputs to this process are the index of the 8x8 luma block luma8x8BlkIdx and variable arrays Intra4x4PredMode and Intra8x8PredMode that are previously (in decoding order) derived for adjacent macroblocks.

Output of this process is the variable Intra8x8PredMode[luma8x8BlkIdx].

Table 8-2 specifies the values for Intra8x8PredMode[luma8x8BlkIdx] and the associated names.

Table 8-2 – Specification of Intra8x8PredMode[luma8x8BlkIdx] and associated names

Intra8x8PredMode[luma8x8BlkIdx]	Name of Intra8x8PredMode[luma8x8BlkIdx]
0	Intra_8x8_Vertical (prediction mode)
1	Intra_8x8_Horizontal (prediction mode)
2	Intra_8x8_DC (prediction mode)
3	Intra_8x8_Diagonal_Down_Left (prediction mode)
4	Intra_8x8_Diagonal_Down_Right (prediction mode)
5	Intra_8x8_Vertical_Right (prediction mode)
6	Intra_8x8_Horizontal_Down (prediction mode)
7	Intra_8x8_Vertical_Left (prediction mode)
8	Intra_8x8_Horizontal_Up (prediction mode)

Let intraMxMPredModeA and intraMxMPredModeB be variables that specify the intra prediction modes of neighbouring 4x4 or 8x8 luma blocks.

Intra8x8PredMode[luma8x8BlkIdx] is derived as follows.
- The process specified in subclause 6.4.7.2 is invoked with luma8x8BlkIdx given as input and the output is assigned to mbAddrA, luma8x8BlkIdxA, mbAddrB, and luma8x8BlkIdxB.
- The variable dcOnlyPredictionFlag is derived as follows.
  - If one of the following conditions is true, dcOnlyPredictionFlag is set equal to 1
    - the macroblock with address mbAddrA is not available
    - the macroblock with address mbAddrB is not available
    - the macroblock with address mbAddrA is available and coded in Inter prediction mode and constrained_intra_pred_flag is equal to 1
    - the macroblock with address mbAddrB is available and coded in Inter prediction mode and constrained_intra_pred_flag is equal to 1
  - Otherwise, dcOnlyPredictionFlag is set equal to 0.
- For N being either replaced by A or B, the variables intraMxMPredModeN are derived as follows.
  - If dcOnlyPredictionFlag is equal to 1 or the macroblock with address mbAddrN is not coded in Intra_4x4 or Intra_8x8 macroblock prediction mode, intraMxMPredModeN is set equal to 2 (Intra_8x8_DC prediction mode).
  - Otherwise (dcOnlyPredictionFlag is equal to 0 and the macroblock with address mbAddrN is coded in Intra_4x4 or Intra_8x8 macroblock prediction mode), the following applies.
    - If the macroblock with address mbAddrN is coded in Intra_8x8 macroblock mode, IntraMxMPredModeN is set equal to Intra8x8PredMode[luma8x8BlkIdxN], where Intra8x8PredMode is the variable array assigned to the macroblock mbAddrN.
    - If the macroblock with address mbAddrN is coded in Intra_4x4 macroblock mode, intraMxMPredModeN is derived by the following procedure, where Intra4x4PredMode is the variable array assigned to the macroblock mbAddrN.

      IntraMxMPredModeA = Intra4x4PredMode [luma8x8BlkIdxA * 4 + 1]
      IntraMxMPredModeB = Intra4x4PredMode [luma8x8BlkIdxB * 4 + 2]

- Intra8x8PredMode[luma8x8BlkIdx] is derived by applying the following procedure.

```
predIntra8x8PredMode = Min( intraMxMPredModeA, intraMxMPredModeB )
if( prev_intra8x8_pred_mode_flag[ luma8x8BlkIdx ] )
    Intra8x8PredMode[ luma8x8BlkIdx ] = predIntra8x8PredMode
else
    if( rem_intra8x8_pred_mode[ luma8x8BlkIdx ] < predIntra8x8PredMode )
        Intra8x8PredMode[ luma8x8BlkIdx ] = rem_intra8x8_pred_mode[ luma8x8BlkIdx ]
    else
        Intra8x8PredMode[ luma8x8BlkIdx ] = rem_intra8x8_pred_mode[ luma8x8BlkIdx ] + 1
```

10197) New subclause 8.3.2.2 "Intra_8x8 sample prediction"

Add a new subclause 8.3.2.2 as follows.

This process is invoked for each 8x8 luma block of a macroblock with prediction mode equal to Intra_8x8 followed by the transform decoding process and picture construction process prior to deblocking for each 8x8 luma block.

Inputs to this process are the index of the 8x8 luma block with index luma8x8BlkIdx and constructed samples prior (in decoding order) to the deblocking filter process from adjacent luma blocks.

Output of this process are the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 for the 8x8 luma block with index luma8x8BlkIdx.

The position of the upper left sample of a 8x8 luma block with index luma8x8BlkIdx inside the current macroblock is derived by invoking the inverse 8x8 luma block scanning process in subclause 6.4.2 with luma8x8BlkIdx as the input and the output being assigned to (xO, yO).

The 25 neighbouring samples p[x, y] that are constructed luma samples prior to the deblocking filter process, with x = −1, y = −1..7 and x = 0..15, y = −1, are derived as follows.
- The luma location (xN, yN) is specified by

xN = xO + x

yN = yO + y

The derivation process for neighbouring locations in subclause 6.4.7 is invoked for luma locations with (xN, yN) as input and mbAddrN and (xW, yW) as output.

Each sample p[x, y] with x = −1, y = −1..7 and x = 0..15, y = −1 is derived as follows.

If any of the following conditions is true, the sample p[x, y] is marked as "not available for Intra_8x8 prediction"

mbAddrN is not available,

the macroblock mbAddrN is coded in Inter prediction mode and constrained_intra_pred_flag is equal to 1.

x is greater than 7 and luma8x8BlkIdx is equal to 3

Otherwise, the sample p[x, y] is marked as "available for Intra_8x8 prediction" and the luma sample at luma location (xW, yW) inside the macroblock mbAddrN is assigned to p[x, y].

When samples p[x, −1], with x = 8..15 are marked as "not available for Intra_8x8 prediction," and the sample p[7, −1] is marked as "available for Intra_8x8 prediction," the sample value of p[7, −1] is substituted for sample values p[x, −1], with x = 7..15 and samples p[x, −1], with x = 7..15 are marked as "available for Intra_8x8 prediction".

NOTE – Each block is assumed to be constructed into a frame prior to decoding of the next block.

Depending on Intra8x8PredMode[luma8x8BlkIdx], one of the Intra_8x8 prediction modes specified in subclauses 8.3.2.2.1 to 8.3.2.2.9 shall be used.

10298) New subclause 8.3.2.2.1 "Specification of Intra_8x8_Vertical prediction mode"

Add a new subclause 8.3.2.2.1 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 0.

This mode shall be used only when the samples p[x, −1] with x = 0..7 are marked as "available for Intra_8x8 prediction".

The values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived by

$pred8x8_L[x, y] = p[x, -1]$, with x, y = 0..7 (Eq. No.)

10399) New subclause 8.3.2.2.2 "Specification of Intra_8x8_Horizontal prediction mode"

Add a new subclause 8.3.2.2.2 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 1.

This mode shall be used only when the samples p[−1, y], with y = 0..7 are marked as "available for Intra_8x8 prediction".

The values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived by

$pred8x8_L[x, y] = p[-1, y]$, with x,y = 0..7 (Eq. No.)

104100) New subclause 8.3.2.2.3"Specification of Intra_8x8_DC prediction mode"

Add a new subclause 8.3.2.2.3 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 2.

The values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived as follows.

If all samples p[x, −1], with x = 0..7 and p[−1, y], with y = 0..7 are marked as "available for Intra_8x8 prediction", the values of the prediction samples $pred8x8_L$[x, y], with x, y = 0..7 are derived by

$$pred8x8_L[x,y] = (p[0,-1] + p[1,-1] + p[2,-1] + p[3,-1] + p[4,-1] + p[5,-1] + p[6,-1] + p[7,-1] + p[-1,0] + p[-1,1] + p[-1,2] + p[-1,3] + p[-1,4] + p[-1,5] + p[-1,6] + p[-1,7] + 8) >> 4 \quad \text{(Eq. No.)}$$

Otherwise, if samples p[x, −1], with x = 0..7 are marked as "not available for Intra_8x8 prediction" and p[−1, y], with y = 0..7 are marked as "available for Intra_8x8 prediction", the values of the prediction samples $pred8x8_L$[x, y], with x, y = 0..7 are derived by

$$pred8x8_L[x,y] = (p[-1,0] + p[-1,1] + p[-1,2] + p[-1,3] + p[-1,4] + p[-1,5] + p[-1,6] + p[-1,7] + 4) >> 3 \quad \text{(Eq. No.)}$$

Otherwise, if samples p[−1, y], with y = 0..7 are marked as "not available for Intra_8x8 prediction" and p[x, −1], with x = 0 .. 7 are marked as "available for Intra_8x8 prediction", the values of the prediction samples $pred8x8_L$[x, y], with x, y = 0 .. 7 are derived by

$$pred8x8_L[x,y] = (p[0,-1] + p[1,-1] + p[2,-1] + p[3,-1] + p[4,-1] + p[5,-1] + p[6,-1] + p[7,-1] + 4) >> 3 \quad \text{(Eq. No.)}$$

Otherwise (all samples p[x, −1], with x = 0..7 and p[−1, y], with y = 0..7 are marked as "not available for Intra_8x8 prediction"), the values of the prediction samples $pred8x8_L$[x, y], with x, y = 0..7 are derived by

$$pred8x8_L[x,y] = 128 \quad \text{(Eq. No.)}$$

NOTE – An 8x8 luma block can always be predicted using this mode.

105101) New subclause 8.3.2.2.4"Specification of Intra_8x8_Diagonal_Down_Left prediction mode"

Add a new subclause 8.3.2.2.4 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 3.

This mode shall be used only when the samples p[x, −1] with x = 0..15 are marked as "available for Intra_8x8 prediction".

The values of the prediction samples $pred8x8_L$[x, y], with x, y = 0..7 are derived as follows.
If x is equal to 7 and y is equal to 7,

$$pred8x8_L[x,y] = (p[14,-1] + 3 * p[15,-1] + 2) >> 2 \quad \text{(Eq. No.)}$$

Otherwise (x is not equal to 7 or y is not equal to 7),

$$pred8x8_L[x,y] = (p[x+y,-1] + 2 * p[x+y+1,-1] + p[x+y+2,-1] + 2) >> 2 \quad \text{(Eq. No.)}$$

106102) New subclause 8.3.2.2.5"Specification of Intra_8x8_Down_Right prediction mode"

Add a new subclause 8.3.2.2.5 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 4.

This mode shall be used only when the samples p[x, −1] with x = 0..7 and p[−1, y] with y = −1..7 are marked as "available for Intra_8x8 prediction".

The values of the prediction samples $pred8x8_L$[x, y], with x, y = 0..7 are derived as follows.

If x is greater than y,

pred8x8$_L$[x, y] = (p[x − y − 2, −1] + 2 * p[x − y − 1, −1] + p[x − y, −1] + 2) >> 2 (Eq. No.)

- Otherwise if x is less than y,

pred8x8$_L$[x, y] = (p[−1, y − x − 2] + 2 * p[−1, y − x − 1] + p[−1, y − x] + 2) >> 2 (Eq. No.)

- Otherwise (x is equal to y),

pred8x8$_L$[x, y] = (p[0, −1] + 2 * p[−1, −1] + p[−1, 0] + 2) >> 2 (Eq. No.)

107103) New subclause 8.3.2.2.6"Specification of Intra_8x8_Vertical_Right prediction mode"

Add a new subclause 8.3.2.2.6 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 5.

This mode shall be used only when the samples p[x, -1] with x = 0..7 and p[-1, y] with y = -1..7 are marked as "available for Intra_8x8 prediction".

Let the variable zVR be set equal to 2 * x − y.

The values of the prediction samples pred8x8$_L$[x, y], with x, y = 0..7 are derived as follows.

- If zVR is equal to 0, 2, 4, 6, 8, 10, 12, or 14

pred8x8$_L$[x, y] = (p[x − (y >> 1) − 1, −1] + p[x − (y >> 1), −1] + 1) >> 1 (Eq. No.)

- Otherwise, if zVR is equal to 1, 3, 5, 7, 9, 11, or 13

pred8x8$_L$[x, y] = (p[x − (y >> 1) − 2, −1] + 2 * p[x − (y >> 1) − 1, −1] + p[x − (y >> 1), −1] + 2) >> 2 (Eq. No.)

- Otherwise, if zVR is equal to -1,

pred8x8$_L$[x, y] = (p[−1, 0] + 2 * p[−1, −1] + p[0, −1] + 2) >> 2 (Eq. No.)

- Otherwise (zVR is equal to -2, -3, -4, -5, -6, or -7),

pred8x8$_L$[x, y] = (p[−1, y − 2*x − 1] + 2 * p[−1, y − 2*x − 2] + p[−1, y − 2*x − 3] + 2) >> 2 (Eq. No.)

108104) New subclause 8.3.2.2.7"Specification of Intra_8x8_Horizontal_Down prediction mode"

Add a new subclause 8.3.2.2.7 as follows.

This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 6.

This mode shall be used only when the samples p[x, −1] with x = 0..7 and p[−1, y] with y = −1..7 are marked as "available for Intra_8x8 prediction".

Let the variable zHD be set equal to 2 * y − x.

The values of the prediction samples pred8x8$_L$[x, y], with x, y = 0..7 are derived as follows.

- If zHD is equal to 0, 2, 4, 6, 8, 10, 12, or 14

pred8x8$_L$[x, y] = (p[−1, y − (x >> 1) − 1] + p[−1, y − (x >> 1)] + 1) >> 1 (Eq. No.)

- Otherwise, if zHD is equal to 1, 3, 5, 7, 9, 11, or 13

~~$pred8x8_L[x, y] = (p[-1, y - (x >> 1) - 2] + 2 * p[-1, y - (x >> 1) - 1] + p[-1, y - (x >> 1)] + 2) >> 2$ (Eq. No.)~~

~~- Otherwise, if zHD is equal to -1,~~

~~$pred8x8_L[x, y] = (p[-1, 0] + 2 * p[-1, -1] + p[0, -1] + 2) >> 2$ (Eq. No.)~~

~~- Otherwise (zHD is equal to -2, -3, -4, -5, -6, -7),~~

~~$pred8x8_L[x, y] = (p[x - 2*y - 1, -1] + 2 * p[x - 2*y - 2, -1] + p[x - 2*y - 3, -1] + 2) >> 2$ (Eq. No.)~~

~~**109105) New subclause 8.3.2.2.8 "Specification of Intra_8x8_Horizontal_Down prediction mode"**~~

~~*Add a new subclause 8.3.2.2.8 as follows.*~~

~~This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 7.~~

~~This mode shall be used only when the samples p[x, -1] with x = 0..15 are marked as "available for Intra_8x8 prediction".~~

~~The values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived as follows.~~

~~- If y is equal to 0, 2, 4 or 6~~

~~$pred8x8_L[x, y] = (p[x + (y >> 1), -1] + p[x + (y >> 1) + 1, -1] + 1) >> 1$ (Eq. No.)~~

~~- Otherwise (y is equal to 1, 3, 5, 7),~~

~~$pred8x8_L[x, y] = (p[x + (y >> 1), -1] + 2 * p[x + (y >> 1) + 1, -1] + p[x + (y >> 1) + 2, -1] + 2) >> 2$ (Eq. No.)~~

~~**110106) New subclause 8.3.2.2.9 "Specification of Intra_8x8_Horizontal_Down prediction mode"**~~

~~*Add a new subclause 8.3.2.2.9 as follows.*~~

~~This Intra_8x8 prediction mode shall be used when Intra8x8PredMode[luma8x8BlkIdx] is equal to 8.~~

~~This mode shall be used only when the samples p[-1, y] with y = 0..7 are marked as "available for Intra_8x8 prediction".~~

~~Let the variable zHU be set equal to x + 2 * y.~~

~~The values of the prediction samples $pred8x8_L[x, y]$, with x, y = 0..7 are derived as follows:~~

~~- If zHU is equal to 0, 2, 4, 6, 8, 10, or 12~~

~~$pred8x8_L[x, y] = (p[-1, y + (x >> 1)] + p[-1, y + (x >> 1) + 1] + 1) >> 1$ (Eq. No.)~~

~~- Otherwise, if zHU is equal to 1, 3, 5, 7, 9, or 11~~

~~$pred8x8_L[x, y] = (p[-1, y + (x >> 1)] + 2 * p[-1, y + (x >> 1) + 1] + p[-1, y + (x >> 1) + 2] + 2) >> 2$ (Eq. No.)~~

~~- Otherwise, if zHU is equal to 13,~~

~~$pred8x8_L[x, y] = (p[-1, 6] + 3 * p[-1, 7] + 2) >> 2$ (Eq. No.)~~

~~- Otherwise (zHU is greater than 13),~~

$pred8x8_L[x, y] = p[-1, 7]$ (Eq. No.)

111107) Subclause 8.5 Transform coefficient decoding process and picture construction process prior to deblocking filter process

Change subclause 8.5 as follows.

Inputs to this process are Intra16x16DCLevel (if available), Intra16x16ACLevel (if available), LumaLevel (if available), LumaLevel64 (if available), ChromaDCLevel, ChromaACLevel, and available Inter or Intra prediction sample arrays for the current macroblock for the applicable component $pred_L$, $pred_{Cb}$, or $pred_{Cr}$.

NOTE – When decoding a macroblock in Intra_4x4 (or Intra_8x8) prediction mode, the luma component of the macroblock prediction array may not be complete, since for each 4x4 (or 8x8) luma block, the Intra_4x4 (or 8x8) prediction process for luma samples as specified in subclause 8.3.1 (or 8.3.2) and the process specified in this subclause are iterated.

NOTE – When decoding a macroblock in Intra_4x4 (or Intra_8x8) prediction mode, the luma component of the macroblock constructed sample arrays prior to the deblocking filter process may not be complete, since for each 4x4 (or 8x8) luma block, the Intra_4x4 (or 8x8) prediction process for luma samples as specified in subclause 8.3.1 (or 8.3.2) and the process specified in this subclause are iterated.

When the current macroblock is coded as P_Skip or B_Skip, all values of LumaLevel, LumaLevel64, ChromaDCLevel, ChromaACLevel are set equal to 0 for the current macroblock.

112108) Subclause 8.5.1 Specification of transform decoding process for 4x4 luma residual blocks

Change subclause 8.5.1 as follows.

Change the title as follows.

"Specification of transform decoding process for 4x4 luma residual blocks"

Add the following paragraph at the start.

This process is invoked for luma 4x4 blocks when TransformSizeIs8x8Flag is equal to 0 or for chroma 4x4 blocks.

113109) New subclause 8.5.2 "Specification of transform decoding process for 8x8 luma residual blocks"

Add a new subclause 8.5.2 as follows.

This process is invoked when TransformSizeIs8x8Flag is equal to 1. The variable LumaLevel64[luma8x8BlkIdx] with luma8x8BlkIdx = 0..3 contains the levels for the luma transform coefficients for the luma 8x8 block with index luma8x8BlkIdx.

For an 8x8 luma block indexed by luma8x8BlkIdx = 0..3, the following ordered steps are specified.

1. The inverse transform coefficient scanning process as described in subclause 8.5.5 is invoked with LumaLevel64[luma8x8BlkIdx] as the input and the two-dimensional array c as the output.
2. The scaling and transformation process for residual 8x8 blocks as specified in subclause 8.5.6 is invoked with c as the input and r as the output.
3. The position of the upper-left sample of an 8x8 luma block with index luma8x8BlkIdx inside the macroblock is derived by invoking the inverse 8x8 luma block scanning process in subclause 6.4.2.2 with luma8x8BlkIdx as the input and the output being assigned to (xO, yO).
4. The 8x8 array u with elements u_{ij} for i, j = 0..7 is derived as

$$u_{ij} = Clip1(pred_L[xO + j, yO + i] + r_{ij}) \quad (8\text{-}243)$$

5.The picture construction process prior to deblocking filter process in subclause 8.5.9 is invoked with luma8x8BlkIdx, u as the input and S' as the output.

114110) Subclause 8.5.4 Inverse scanning process for transform coefficients

Change the title as follows.

"Inverse scanning process for 4x4 transform coefficients"

115111) New subclause 8.5.5 "Inverse scanning process for 8x8 luma transform coefficients"

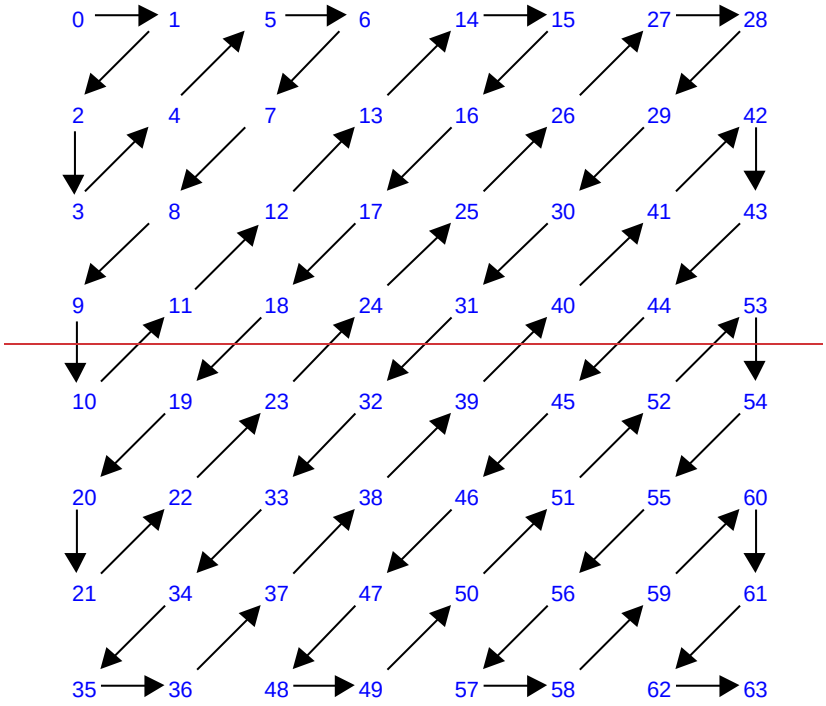
Add a new subclause 8.5.5 as follows.

Input to this process is a list of 64 values.

Output of this process is a variable c containing a two-dimensional array of 8x8 values with level assigned to locations in the transform block.

The decoding process maps the sequence of transform coefficient levels to the transform coefficient level positions. For this mapping, the two inverse scanning patterns shown in Figure 8-9 are used.

The inverse zig-zag scan shall be used for frame macroblocks and the inverse field scan shall be used for field macroblocks.



a) Zig-zag scan

0	3	8	15	22	30	38	52
1	4	14	21	29	37	45	53
2	7	16	23	31	39	46	58
5	9	20	28	36	44	51	59
6	13	24	32	40	47	54	60
10	17	25	33	41	48	55	61
11	18	26	34	42	49	56	62
12	19	27	35	43	50	57	63

b) Field scan [Ed. Note (TW): convert to visio drawing]

Figure 8-9 – 8x8 Scan Orders

Table 8-13 provides the mapping from the index idx of input list of 64 elements to indices i and j of the two-dimensional array c.

Table 8-13 – Specification of mapping of idx to c_{ij} for zig-zag and field scan

idx	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
zig-zag	c_{00}	c_{01}	c_{10}	c_{20}	c_{11}	c_{02}	c_{03}	c_{12}	c_{21}	c_{30}	c_{40}	c_{31}	c_{22}	c_{13}	c_{04}	c_{05}
field	c_{00}	c_{10}	c_{20}	c_{01}	c_{11}	c_{30}	c_{40}	c_{21}	c_{02}	c_{31}	c_{50}	c_{60}	c_{70}	c_{41}	c_{12}	c_{03}

Table 8-13 – Specification of mapping of idx to cij for zig-zag and field scan (concluded)

idx	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
zig-zag	c_{14}	c_{23}	c_{32}	c_{41}	c_{50}	c_{60}	c_{51}	c_{42}	c_{33}	c_{24}	c_{15}	c_{06}	c_{07}	c_{16}	c_{25}	c_{34}
field	c_{22}	c_{51}	c_{61}	c_{71}	c_{32}	c_{13}	c_{04}	c_{23}	c_{42}	c_{52}	c_{62}	c_{72}	c_{33}	c_{14}	c_{05}	c_{24}

Table 8-13 – Specification of mapping of idx to cij for zig-zag and field scan (concluded)

idx	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
zig-zag	c_{43}	c_{52}	c_{61}	c_{70}	c_{71}	c_{62}	c_{53}	c_{44}	c_{35}	c_{26}	c_{17}	c_{27}	c_{36}	c_{45}	c_{54}	c_{63}
field	c_{43}	c_{53}	c_{63}	c_{73}	c_{34}	c_{15}	c_{06}	c_{25}	c_{44}	c_{54}	c_{64}	c_{74}	c_{35}	c_{16}	c_{26}	c_{45}

Table 8-13 – Specification of mapping of idx to cij for zig-zag and field scan (concluded)

idx	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
zig-zag	c_{72}	c_{73}	c_{64}	c_{55}	c_{46}	c_{37}	c_{47}	c_{56}	c_{65}	c_{74}	c_{75}	c_{66}	c_{57}	c_{67}	c_{76}	c_{77}
field	c_{55}	c_{65}	c_{75}	c_{36}	c_{07}	c_{17}	c_{46}	c_{56}	c_{66}	c_{76}	c_{27}	c_{37}	c_{47}	c_{57}	c_{67}	c_{77}

116112) Subclause 8.5.5 "Derivation process for the quantisation parameters and scaling function"

Add to subclause 8.5.5 the following.

The function LevelScale64(m, i, j) is specified as follows:

$$\text{LevelScale64}(m,i,j) = \begin{cases} V_{m0} & \text{for } (i,j) \text{ with } i \in \{0,4\},\ j \in \{0,4\}, \\ V_{m1} & \text{for } (i,j) \text{ with } i \in \{1,3,5,7\},\ j \in \{1,3,5,7\}, \\ V_{m2} & \text{for } (i,j) \text{ with } i \in \{2,6\},\ j \in \{2,6\}, \\ V_{m3} & \text{for } (i,j) \text{ with } (i \in \{0,4\},\ j \in \{1,3,5,7\}) \cup (i \in \{1,3,5,7\},\ j \in \{0,4\}), \\ V_{m4} & \text{for } (i,j) \text{ with } (i \in \{0,4\},\ j \in \{2,6\}) \cup (i \in \{2,6\},\ j \in \{0,4\}), \\ V_{m5} & \text{otherwise;} \end{cases}$$

(Eq. No.)

where the first and second subscripts of V are row and column indices, respectively, of the matrix specified as:

$$V = \begin{bmatrix} 20 & 18 & 32 & 19 & 25 & 24 \\ 22 & 19 & 35 & 21 & 28 & 26 \\ 26 & 23 & 42 & 24 & 33 & 31 \\ 28 & 25 & 45 & 26 & 35 & 33 \\ 32 & 28 & 51 & 30 & 40 & 38 \\ 36 & 32 & 58 & 34 & 46 & 43 \end{bmatrix}$$ (Eq. No.)

117113) New subclause 8.5.10 "Scaling and transformation process for residual 8x8 blocks"

Add a new subclause 8.5.10 as follows.

Input to this process is an 8x8 array c with elements c_{ij} which is an array relating to an 8x8 residual block of the luma component.

Outputs of this process are residual sample values as 8x8 array r with elements r_{ij}.

Scaling of 8x8 block transform coefficient levels c_{ij} proceeds as follows.

If QP_Y is greater than or equal to 12, scaling of 8x8 block transform coefficient levels c_{ij} shall be performed as

$$d_{ij} = (c_{ij} * \text{LevelScale64}(QP_Y \%6, i, j)) << (QP_Y / 6 - 2), \quad \text{with} \quad i, j = 0..7$$

Otherwise (QP_Y is less than 12), scaling of 8x8 block transform coefficient levels c_{ij} shall be performed as

$$d_{ij} = (c_{ij} * \text{LevelScale64}(QP_Y \%6, i, j) + 2^{1-QP_Y/6}) >> (2 - QP_Y / 6), \quad \text{with} \quad i, j = 0..7$$

The transform process shall convert the block of scaled transform coefficients to a block of output samples in a manner mathematically equivalent to the following.

First, each (horizontal) row of scaled transform coefficients is transformed using a one-dimensional inverse transform as follows.

A set of intermediate values e_{ij} is computed as follows.

$$e_{i0} = d_{i0} + d_{i4}, \quad \text{with} \quad i = 0..7$$ (Eq. No.)

$$e_{i1} = -d_{i3} + d_{i5} - d_{i7} - (d_{i7} >> 1), \quad \text{with} \quad i = 0..7$$ (Eq. No.)

$$e_{i2} = d_{i0} - d_{i4}, \quad \text{with} \quad i = 0..7$$ (Eq. No.)

$$e_{i3} = d_{i1} + d_{i7} - d_{i3} - (d_{i3} >> 1), \quad \text{with} \quad i = 0..7$$ (Eq. No.)

$e_{i4} = (d_{i2} >> 1) - d_{i6}$, with $i = 0..7$ (Eq. No.)

$e_{i5} = -d_{i1} + d_{i7} + d_{i5} + (d_{i5} >> 1)$, with $i = 0..7$ (Eq. No.)

$e_{i6} = d_{i2} + (d_{i6} >> 1)$, with $i = 0..7$ (Eq. No.)

$e_{i7} = d_{i3} + d_{i5} + d_{i1} + (d_{i1} >> 1)$, with $i = 0..7$ (Eq. No.)

A second set of intermediate results f_{ij} is computed from the intermediate values e_{ij} as follows.

$f_{i0} = e_{i0} + e_{i6}$, with $i = 0..7$ (Eq. No.)

$f_{i1} = e_{i1} + (e_{i7} >> 2)$, with $i = 0..7$ (Eq. No.)

$f_{i2} = e_{i2} + e_{i4}$, with $i = 0..7$ (Eq. No.)

$f_{i3} = e_{i3} + (e_{i5} >> 2)$ with $i = 0..7$ (Eq. No.)

$f_{i4} = e_{i2} - e_{i4}$, with $i = 0..7$ (Eq. No.)

$f_{i5} = (e_{i3} >> 2) - e_{i5}$ with $i = 0..7$ (Eq. No.)

$f_{i6} = e_{i0} - e_{i6}$, with $i = 0..7$ (Eq. No.)

$f_{i7} = e_{i7} - (e_{i1} >> 2)$, with $i = 0..7$ (Eq. No.)

Then, the transformed result g_{ij} is computed from these intermediate values f_{ij} as follows.

$g_{i0} = f_{i0} + f_{i7}$, with $i = 0..7$ (Eq. No.)

$g_{i1} = f_{i2} + f_{i5}$, with $i = 0..7$ (Eq. No.)

$g_{i2} = f_{i4} + f_{i3}$, with $i = 0..7$ (Eq. No.)

$g_{i3} = f_{i6} + f_{i1}$, with $i = 0..7$ (Eq. No.)

$g_{i4} = f_{i6} - f_{i1}$, with $i = 0..7$ (Eq. No.)

$g_{i5} = f_{i4} - f_{i3}$, with $i = 0..7$ (Eq. No.)

$g_{i6} = f_{i2} - f_{i5}$, with $i = 0..7$ (Eq. No.)

$g_{i7} = f_{i0} - f_{i7}$, with $i = 0..7$ (Eq. No.)

Then, each (vertical) column of the resulting matrix is transformed using the same one-dimensional inverse transform as follows.

A set of intermediate values h_{ij} is computed from the horizontally transformed value g_{ij} as follows.

$h_{i0} = g_{i0} + g_{i4}$, with i = 0..7 (Eq. No.)

$h_{i1} = g_{i3} + g_{i5} - g_{i7} - (g_{i7} >> 1)$, with i = 0..7 (Eq. No.)

$h_{i2} = g_{i0} - g_{i4}$, with i = 0..7 (Eq. No.)

$h_{i3} = g_{i1} + g_{i7} - g_{i3} - (g_{i3} >> 1)$, with i = 0..7 (Eq. No.)

$h_{i4} = (g_{i2} >> 1) - g_{i6}$, with i = 0..7 (Eq. No.)

$h_{i5} = -g_{i1} + g_{i7} + g_{i5} + (g_{i5} >> 1)$, with i = 0..7 (Eq. No.)

$h_{i6} = g_{i2} + (g_{i6} >> 1)$, with i = 0..7 (Eq. No.)

$h_{i7} = g_{i3} + g_{i5} + g_{i1} + (g_{i1} >> 1)$, with i = 0..7 (Eq. No.)

A second set of intermediate results k_{ij} is computed from the intermediate values h_{ij} as follows.

$k_{i0} = h_{i0} + h_{i6}$, with i = 0..7 (Eq. No.)

$k_{i1} = h_{i1} + (h_{i7} >> 2)$, with i = 0..7 (Eq. No.)

$k_{i2} = h_{i2} + h_{i4}$, with i = 0..7 (Eq. No.)

$k_{i3} = h_{i3} + (h_{i5} >> 2)$ with i = 0..7 (Eq. No.)

$k_{i4} = h_{i2} - h_{i4}$, with i = 0..7 (Eq. No.)

$k_{i5} = (h_{i3} >> 2) - h_{i5}$ with i = 0..7 (Eq. No.)

$k_{i6} = h_{i0} - h_{i6}$, with i = 0..7 (Eq. No.)

$k_{i7} = h_{i7} - (h_{i1} >> 2)$, with i = 0..7 (Eq. No.)

Then, the transformed result m_{ij} is computed from these intermediate values k_{ij} as follows.

$m_{i0} = k_{i0} + k_{i7}$, with i = 0..7 (Eq. No.)

$m_{i1} = k_{i2} + k_{i5}$, with i = 0..7 (Eq. No.)

$m_{i2} = k_{i4} + k_{i3}$, with i = 0..7 (Eq. No.)

$m_{i3} = k_{i6} + k_{i1}$, with i = 0..7 (Eq. No.)

$m_{i4} = k_{i6} - k_{i1}$, with i = 0..7 (Eq. No.)

$m_{i5} = k_{i4} - k_{i3}$, with i = 0..7 (Eq. No.)

$m_{i6} = k_{i2} - k_{i5}$, with i = 0..7 (Eq. No.)

$m_{i7} = k_{i0} - k_{i7}$, with i = 0..7 (Eq. No.)

After performing both the one-dimensional horizontal and the one-dimensional vertical inverse transforms to produce an array of transformed samples, the final constructed residual sample values shall be derived as

$r_{ij} = (m_{ij} + 2^5) >> 6$ with i, j = 0..7 (Eq. No.)

118114) Subclause 8.5.10Picture construction process prior to deblocking filter process

Change the title as follows.

"Picture construction process for 4x4 blocks prior to deblocking filter process"

119115) New subclause 8.5.12 "Picture construction process for 8x8 luma residuals prior to deblocking filter process"

Add a new subclause 8.5.12 as follows.

Inputs to this process are

- luma8x8BlkIdx
- a constructed residual sample 8x8 array u with elements u_{ij} which is a luma residual block
- the prediction sample 8x8 array $pred_L$

Outputs of this process are constructed luma sample blocks S'_L prior to the deblocking filter process.

The position of the upper left luma sample of the current macroblock is derived by invoking the inverse macroblock scanning process in subclause 6.4.1 with CurrMbAddr as input and the output being assigned to (xP, yP).

For each sample u_{ij} of the 8x8 luma block, the following applies.

- The position of the upper-left sample of an 8x8 luma block with index luma8x8BlkIdx inside the macroblock is derived by invoking the inverse 8x8 luma block scanning process in subclause 6.4.2.2 with luma8x8BlkIdx as the input and the output being assigned to (xO, yO).
- Depending on the variable MbaffFrameFlag, the following applies.
  - If MbaffFrameFlag is equal to 1 and the current macroblock is a field macroblock

    $S'_L[xP + xO + j, yP + 2 * (yO + i)] = u_{ij}$ with i, j = 0..7 (Eq. No)

  - Otherwise (MbaffFrameFlag is equal to 0 or the current macroblock is a frame macroblock),

    $S'_L[xP + xO + j, yP + yO + i] = u_{ij}$ with i, j = 0..7 (Eq. No)

120116) Subclause 8.7 "Deblocking filter process"

Change subclause 8.7 as follows.

Change the first sentence of the first paragraph as follows.

A conditional filtering shall be applied to all 4x4 NxN (where N = 4 or N = 8 for luma, and N = 4 for chroma) block edges of a picture, except edges at the boundary of the picture and any edges for which the deblocking filter process is disabled by disable_deblocking_filter_idc, as specified below.

Change the second sentence of the second paragraph as follows.

The deblocking filter process is invoked for the luma and chroma components separately. For each macroblock, vertical edges are filtered first, from left to right, and then horizontal edges are filtered from top to bottom. The luma deblocking filter process is performed on two (when N = 8) or four (when N = 4) 16-sample edges and the deblocking filter process for each chroma components is performed on two 8-sample edges, for the horizontal direction as shown on the left side of Figure 8-9 and for the vertical direction as shown on the right side of Figure 8-9.#

Change Figure 8-9 as follows.

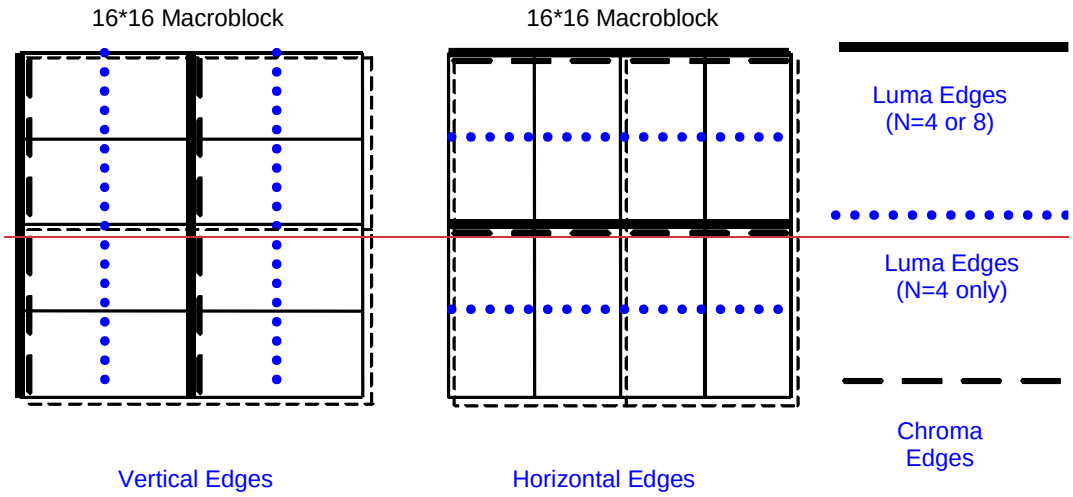


Figure 8-9 Boundaries in a macroblock to be filtered (luma boundaries shown with solid lines for N=4 and N=8, luma boundaries shown with dotted lines for N=4 and chroma boundaries shown with dashed lines)

Change the text below Figure 8-9 as follows.

For each macroblock in ascending order of mbAddr, the following applies.

1.The variables fieldModeMbFlag, filterNon8x8LumaEdgesFlag, filterInternalEdgesFlag, filterLeftMbEdgeFlag and filterTopMbEdgeFlag are derived as follows.

- The variable fieldModeMbFlag is derived as follows.
  - If any of the following conditions is true, fieldModeMbFlag is set equal to 1.
    - field_pic_flag is equal to 1
    - MbaffFrameFlag is equal 1 and the macroblock mbAddr is a field macroblock
  - Otherwise, fieldModeMbFlag is set equal to 0.
- The variable filterNon8x8LumaEdgesFlag is derived as follows.
  - If any of the following conditions is true, filterNon8x8LumaEdgesFlag is set equal to 1.
    - transform_8x8_mode_flag is equal to 0
    - NoMbPartLessThan8x8Flag is equal to 0

- transform_size_flag is equal to 0
- mb_type for the macroblock mbAddr is equal to Intra_16x16

- Otherwise, filterNon8x8LumaEdgesFlag is set equal to 0.

The variable filterInternalEdgesFlag is derived as follows.

- If disable_deblocking_filter_idc for the slice that contains the macroblock mbAddr is equal to 1, the variable filterInternalEdgesFlag is set equal to 0;
- Otherwise (disable_deblocking_filter_idc for the slice that contains the macroblock mbAddr is not equal to 1), the variable filterInternalEdgesFlag is set equal to 1.

The variable filterLeftMbEdgeFlag is derived as follows.

If any of the following conditions is true, the variable filterLeftMbEdgeFlag is set equal to 0.

- the left vertical macroblock edge of the macroblock mbAddr represents a picture boundary
- disable_deblocking_filter_idc for the slice that contains the macroblock mbAddr is equal to 1
- disable_deblocking_filter_idc for the slice that contains the macroblock mbAddr is equal to 2 and the left vertical macroblock edge of the macroblock mbAddr represents a slice boundary

Otherwise, the variable filterLeftMbEdgeFlag is set equal to 1.

The variable filterTopMbEdgeFlag is derived as follows.

If any of the following conditions is true, the variable filterTopMbEdgeFlag is set equal to 0.

- the top horizontal macroblock edge of the macroblock mbAddr represents a picture boundary
- disable_deblocking_filter_idc for the slice that contains the macroblock mbAddr is equal to 1
- disable_deblocking_filter_idc for the slice that contains the macroblock mbAddr is equal to 2 and the top horizontal macroblock edge of the macroblock mbAddr represents a slice boundary

Otherwise, the variable filterTopMbEdgeFlag is set equal to 1.

2.Given the variables fieldModeMbFlag, filterNon8x8LumaEdgesFlag, filterInternalEdgesFlag, filterLeftMbEdgeFlag and filterTopMbEdgeFlag the deblocking filtering is controlled as follows.

When filterLeftMbEdgeFlag is equal to 1, the filtering of the left vertical luma edge is specified as follows.

The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (0, k)$ with k = 0..15 as input and S'_L as output.

When filterInternalEdgesFlag is equal to 1, the filtering of the internal vertical luma edges is specified as follows.

If filterNon8x8LumaEdgesFlag is equal to 1, Tthe process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (4, k)$ with k = 0..15 as input and S'_L as output.

The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (8, k)$ with k = 0..15 as input and S'_L as output.

If filterNon8x8LumaEdgesFlag is equal to 1, Tthe process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (12, k)$ with k = 0..15 as input and S'_L as output.

When filterTopMbEdgeFlag is equal to 1, the filtering of the top horizontal luma edge is specified as follows.

If MbaffFrameFlag is equal to 1, (mbAddr % 2) is equal to 0, mbAddr is greater than or equal to 2 * PicWidthInMbs, the macroblock mbAddr is a frame macroblock, and the macroblock (mbAddr − 2 * PicWidthInMbs + 1) is a field macroblock, the following applies.

- The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = 1, and $(xE_k, yE_k) = (k, 0)$ with k = 0..15 as input and S'_L as output.
- The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = 1, and $(xE_k, yE_k) = (k, 1)$ with k = 0..15 as input and S'_L as output.

- Otherwise, the process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (k, 0)$ with k = 0..15 as input and S'_L as output.

- When filterInternalEdgesFlag is equal to 1, the filtering of the internal horizontal luma edges is specified as follows.

  - If filterNon8x8LumaEdgesFlag is equal to 1, Tthe process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (k, 4)$ with k = 0..15 as input and S'_L as output.

  - The process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (k, 8)$ with k = 0..15 as input and S'_L as output.

  - If filterNon8x8LumaEdgesFlag is equal to 1, TtThe process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (k, 12)$ with k = 0..15 as input and S'_L as output.

121117) Subclause 9.3.1.1 "Initialisation process for context variables"

Change subclause 9.3.1.1 as follows.

Change Table 9-11 with the following.

	Syntax element	Table	Slice type			
			SI	I	P, SP	B
slice_data( )	mb_skip_flag	Table 9-13 Table 9-14			11-13	24-26
	mb_field_decoding_flag	Table 9-18	70-72	70-72	70-72	70-72
macroblock_layer( )	mb_type	Table 9-12, Table 9-13, Table 9-14.	0-10	3-10	14-20	27-35
	transform_size_flag	Table 9-16	na	399-401	399-401	399-401
	coded_block_pattern (luma)	Table 9-18	73-76	73-76	73-76	73-76
	coded_block_pattern (chroma)	Table 9-18	77-84	77-84	77-84	77-84
	mb_qp_delta	Table 9-17	60-63	60-63	60-63	60-63
mb_pred( )	prev_intra4x4_pred_mode_flag	Table 9-17	68	68	68	68
	rem_intra4x4_pred_mode	Table 9-17	69	69	69	69
	prev_intra8x8_pred_mode_flag	Table 9-17	na	68	68	68
	rem_intra8x8_pred_mode	Table 9-17	na	69	69	69
	intra_chroma_pred_mode	Table 9-17	64-67	64-67	64-67	64-67
mb_pred( ) and sub_mb_pred( )	ref_idx_l0	Table 9-16			54-59	54-59
	ref_idx_l1	Table 9-16				54-59
	mvd_l0[ ][ ][0]	Table 9-15			40-46	40-46
	mvd_l1[ ][ ][0]	Table 9-15				40-46
	mvd_l0[ ][ ][1]	Table 9-15			47-53	47-53
	mvd_l1[ ][ ][1]	Table 9-15				47-53

sub_mb_pred( )	sub_mb_type	Table 9-13 Table 9-14			21-23	36-39
residual_block_cabac( )	coded_block_flag	Table 9-18	85-104	85-104	85-104	85-104
	significant_coeff_flag[ ]	Table 9-19, Table 9-22, Table 9-XX, Table 9-XX	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450
	last_significant_coeff_flag[ ]	Table 9-20, Table 9-23, Table 9-XX, Table 9-XX.	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459
	coeff_abs_level_minus1[ ]	Table 9-21, Table 9-XX	227-275 426-435	227-275 426-435	227-275 426-435	227-275 426-435

Change Table 9-16 as follows.

Table 9-16 – Values of variables m and n for ctxIdx from 54 to 59, and 399 to 401 [Ed. (DM): initialization values for two cabac_init_idc cases may be changed]

Value of cabac_init_idc	Initialisation variables	ctxIdx								
		54	55	56	57	58	59	399	400	401
I slices	m	na	na	na	na	na	na	0	0	0
	n	na	na	na	na	na	na	41	63	63
0	m	-7	-5	-4	-5	-7	1	0	0	0
	n	67	74	74	80	72	58	41	63	63
1	m	-1	-1	1	-2	-5	0	0	0	0
	n	66	77	70	86	72	61	41	63	63
2	m	3	-4	-2	-12	-7	1	0	0	0
	n	55	79	75	97	50	60	41	63	63

Insert a Table as follows.

Table 9-XX – Values of variables m and n for ctxIdx from 402 to 459 [Ed. (DM): initialization values for two cabac_init_idc cases may be further refined]

ctxIdx	I slices		Value of cabac_init_idc						ctxIdx	I slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
402	-1	73	-4	60	-4	60	-4	60	431	-4	56	-7	54	-7	54	-7	54
403	-7	73	-3	49	-3	49	-3	49	432	-1	59	-2	58	-2	58	-2	58
404	-6	76	-2	50	-2	50	-2	50	433	-6	71	-4	63	-4	63	-4	63
405	-7	71	-4	49	-4	49	-4	49	434	-8	74	-5	66	-5	66	-5	66
406	-9	72	-5	48	-5	48	-5	48	435	-11	85	1	64	1	64	1	64
407	-5	65	-2	46	-2	46	-2	46	436	-1	73	-4	60	-4	60	-4	60
408	-14	83	-7	54	-7	54	-7	54	437	-7	73	-3	49	-3	49	-3	49
409	-8	72	-1	45	-1	45	-1	45	438	-6	76	-2	50	-2	50	-2	50
410	-10	75	-4	49	-4	49	-4	49	439	-7	71	-4	49	-4	49	-4	49
411	-5	64	4	39	4	39	4	39	440	-9	72	-5	48	-5	48	-5	48
412	-4	59	0	42	0	42	0	42	441	-5	65	-2	46	-2	46	-2	46
413	-13	79	2	43	2	43	2	43	442	-14	83	-7	54	-7	54	-7	54
414	-9	69	0	44	0	44	0	44	443	-8	72	-1	45	-1	45	-1	45
415	-8	66	5	32	5	32	5	32	444	-10	75	-4	49	-4	49	-4	49
416	3	55	15	30	15	30	15	30	445	-5	64	4	39	4	39	4	39
417	12	33	17	27	17	27	17	27	446	-4	59	0	42	0	42	0	42
418	5	38	23	13	23	13	23	13	447	-13	79	2	43	2	43	2	43
419	9	34	24	16	24	16	24	16	448	-9	69	0	44	0	44	0	44
420	18	22	22	25	22	25	22	25	449	-8	66	5	32	5	32	5	32
421	19	22	23	27	23	27	23	27	450	3	55	15	30	15	30	15	30
422	23	19	23	32	23	32	23	32	451	12	33	17	27	17	27	17	27
423	26	16	17	43	17	43	17	43	452	5	38	23	13	23	13	23	13
424	14	44	17	49	17	49	17	49	453	9	34	24	16	24	16	24	16
425	40	14	2	70	2	70	2	70	454	18	22	22	25	22	25	22	25
426	-9	75	-3	58	-3	58	-3	58	455	19	22	23	27	23	27	23	27
427	-1	44	-1	28	-1	28	-1	28	456	23	19	23	32	23	32	23	32
428	-2	49	0	29	0	29	0	29	457	26	16	17	43	17	43	17	43
429	-2	51	2	30	2	30	2	30	458	14	44	17	49	17	49	17	49
430	-1	51	1	35	1	35	1	35	459	40	14	2	70	2	70	2	70

~~**122118) Subclause 9.3.2 "Binarization Process"**~~

~~*Change the following praragraph in subclause 9.3.2 as follows.*~~

~~The possible values of the context index ctxIdx are in the range 0 to 398459, inclusive. The value assigned to ctxIdxOffset specifies the lower value of the range of ctxIdx assigned to the corresponding binarization or binarization part of a syntax element.~~